(19) KOREAN INTELLECTUAL PROPERTY OFFICE

KOREAN PATENT ABSTRACTS

(11)Publication number: 1020020017669 A
(43)Date of publication of application: 07.03.2002

(21)Application number: 1020000051156
(22)Date of filing: 31.08.2000

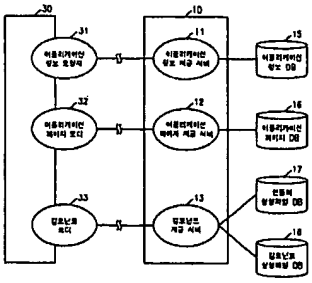
(51)Int. Cl G06F 17/00

(71)Applicant: YANG, JU HO
YANG, JUN HO

(72)Inventor: YANG, JU HO
YANG, JUN HO

(54) SYSTEM FOR OFFERING XML BASED APPLICATION AND ITS COMPUTER READABLE MEDIA

(57) Abstract:



PURPOSE: A XML based application offering system and its computer readable media is provided to install only a container at a client system, and to store remaining necessary files at a server for easily updating or modifying offered files at the client system, and to set an access authority at an application page for efficiently performing a security management.

CONSTITUTION: The system comprises an application data storage(15), an application data offering server(11), an application page storage(16), an application page offering server(12), a handler execution file storage(17), a component execution file storage(18), and a component offering server(13). The application data storage(15) includes a plurality of components, a CTD(Component Type Definition) document written as a DTD(Data Type Definition) for recording registration data on a handler to handle the components, and a list of application pages where the components and handler are declared. The application data offering server(11) offers the application data according to a request from the container. The application page storage(16) stores the application pages. The application page offering server(12) offers the application pages according to a request from the container based on the list of the application pages. The storage(17) stores the handler execution file for executing defined in the CTD document. The storage(18) stores the file for executing the component defined at the CTD document. The component offering server(13) offers the handler execution file according to a request from the container, and also offers component execution files according to request from the activated handler.

Legal Status
Date of final disposal of an application (20040329)

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.		(11) 공개번호	특2002-0017669
G06F 17/00		(43) 공개일자	2002년03월07일
(21) 출원번호	10-2000-0051156		
(22) 출원일자	2000년08월31일		

(71) 출원인	양준호 대한민국 151-850 서울 관악구 봉천11동 1640-3 양주호 대한민국 151-850 서울 관악구 봉천11동 1640-3
(72) 발명자	양준호 대한민국 151-850 서울 관악구 봉천11동 1640-3 양주호 대한민국 151-850 서울 관악구 봉천11동 1640-3
(74) 대리인	이영필 최흥수 이해영
(77) 심사청구	있음

(54) 출원명	XML 기반 어플리케이션 제공 시스템, 및 XML 기반어플리케이션이 기록된 컴퓨터 판독가능한 기록매체

요약

본 발명은 XML 기반 어플리케이션 제공 시스템 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체에 관한 것이다.

본 발명에 따른 XML 기반 어플리케이션 제공 시스템은, 복수의 컴포넌트, 및 상기 컴포넌트를 핸들링하기 위한 핸들러에 대한 등록정보 문서로서 DTD(Data Type Definition)의 문법에 따라 작성된 CTD(Component Type Definition) 문서; 및 상기 CTD문서에 등록된 컴포넌트 및 핸들러가 선언된 어플리케이션 페이지의 리스트를 포함하는 어플리케이션 정보가 저장된 어플리케이션 정보 저장부; 고객의 컴퓨터에 탑재된 컨테이너의 요청에 따라, 상기 어플리케이션 정보 저장부에 저장된 어플리케이션 정보를 제공하기 위한 어플리케이션 정보 제공 서버; 상기 어플리케이션 페이지가 저장된 어플리케이션 페이지 저장부; 상기 제공받은 어플리케이션 페이지의 리스트에 기초한 상기 컨테이너의 요청에 따라 상기 어플리케이션 페이지 저장부에 저장된 어플리케이션 페이지를 제공하기 위한 어플리케이션 페이지 제공 서버; 상기 CTD 문서에 정의된 핸들러를 실행하기 위한 핸들러 실행파일이 저장된 핸들러 실행파일 저장부; 상기 CTD 문서에 정의된 컴포넌트를 실행하기 위한 실행파일이 저장된 컴포넌트 실행파일 저장부; 및 상기 제공받은 어플리케이션 페이지를 실행하기 위한 상기 컨테이너의 요청에 따라 상기 핸들러 실행파일 저장부에 저장된 핸들러 실행파일을 제공하며, 상기 핸들러 실행파일이 실행됨에 따라 활성화된 핸들러의 요청에 따라 상기 컴포넌트 실행파일 저장부에 저장된 대응 컴포넌트 실행파일을 제공하기 위한 컴포넌트 제공 서버를 포함하는 것을 특징으로 한다. 이에 의해, XML을 기반으로 프로그램된 어플리케이션을 제공할 수 있는 XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체가 제공된다.

대표도

도3

명세서

도면의 간단한 설명

도 1은 XML 기반 어플리케이션 제공 시스템의 제1 실시예에 따른 XML 기반 어플리케이션 제공 서버 시스템의 적용도,

도 2는 본 발명에 따른 XML 기반 어플리케이션의 구조를 설명하기 위한 참고도,

도 3은 도 1의 고객 컴퓨터(3a,3n) 및 XML 기반 어플리케이션 제공 서버(1)의 블록도,

도 4는 본 발명의 제2 실시예에 따른 XML 기반 어플리케이션이 기록된 기록매체가 탑재된 컴퓨터(43)의 개략도,

도 5는 본 발명의 XML 기반 어플리케이션 제공 시스템의 제 3실시예로서, XML 기반 어플리케이션을 실행하기 위한 적어도 하나의 컴퓨터(53a, 53n)가 연결된 네트워크 구성도,

도 6은 도 4의 컴퓨터(43), 또는 도 5의 컴퓨터(53a,53n) 및 XML 기반 어플리케이션 제공 데이터베이스(1000))의 블록도,

도 7은 어플리케이션 제공방법에 의해 본 발명에 따른 XML 기반 어플리케이션이 실행되는 과정을 설명하기 위한 참고도이다.

* 도면의 주요 부분에 대한 부호의 설명

1: XML 기반 어플리케이션 제공 시스템∘3a,3n: 고객 컴퓨터

5: 인터넷/인트라넷∘∘ ∘10: 서버부

11: 어플리케이션 제공 서버∘∘12: 어플리케이션 페이지 제공 서버

13: 컴포넌트 제공 서버∘∘∘15: 어플리케이션 정보 데이터베이스

16: 어플리케이션 페이지 데이터베이스

17: 핸들러 실행파일 데이터베이스

18: 컴포넌트 실행파일 데이터베이스

31: 어플리케이션 정보 요청자∘32: 어플리케이션 페이지 로더

33: 컴포넌트 로더∘∘∘∘50: LAN/WAN

53a,53n: 고객 컴퓨터∘∘∘100: 컴퓨터 본체

101: 디스플레이 장치∘∘∘102: 키보드

103: 마우스

1000: XML 기반 어플리케이션 제공 데이터베이스

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은, XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체에 관한 것으로, 보다 상세하게는, XML문법구조를 이용하여 코딩된 어플리케이션을 제공하기 위한 XML 기반 어플리케이션 제공 시스템 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체에 관한 것이다.

어플리케이션이란 사용자 또는 타 응용프로그램에 대해 특정기능을 직접 수행하도록 설계된 프로그램을 말한다. 응용프로그램의 예로는 워드프로세서, 데이터베이스 프로그램, 개발도구, 페인트 브러시, 이미지 편집 프로그램, 통신 프로그램 등을 들 수 있다.

종래 개발자들은 어플리케이션을 개발하기 위해 C++, Java, Basic, 등과 같은 고급 프로그래밍 언어를 사용해왔다. 그러나, 이와 같은 종래 개발언어 및 도구를 사용하여 어플리케이션을 코딩하는 것은 용이하지 않다. 즉, 어플리케이션을 코딩하기 위해서는 코딩 양이 적지 않아 유능한 개발자라 하더라도 상당한 시간이 필요하게 된다. 더욱이, 코딩양이 많을수록 버그발생율이 상대적으로 높아지며 이에 따른 디버깅작업 또한 쉽지 않다. 나아가, 어플리케이션을 업데이트하기 위해서는 소스코드를 검색하고 수정 또는 대체해야 하는 바, 코딩양이 많을수록 이를 위해 더 많은 시간과 노력이 소요된다.

최근 인터넷을 기반으로 하여 ASP(Application Service Provider)가 활성화되고 있다. 이는 단순히 어플리케이션 소프트웨어를 패키지화하여 판매하는 것이 아니라, 비용을 지불하고 일정 기간 동안 어플리케이션을 사용할 수 있게 하는 일종의 어플리케이션 아웃소싱이다. 그런데, 특히 ASP사업을 함에 있어서 대부분의 비용은 전술한 바대로, 개발, 디버깅 및 수정작업에 소요되고 있는 실정이다.

한편, XML(eXtensible Markup Language)은 1996년 W3C(World Wide Web Consortium)에서 제안한 마크업언어로서, 웹 상에서 구조화된 문서를 전송 가능하도록 설계된 표준화된 텍스트 포맷을 제공한다. XML은 기존에 사용하던 HTML의 한계를 극복하고 SGML의 복잡함을 해결하는 방안의 하나로, SGML의 실용적인 기능만을 모은 부분집합(subset)이며, HTML과 달리 사용자가 새로운 태그(tag)를 정의할 수 있는 것이 큰 특징이다.

HTML(HyperText Markup Language)은 하이퍼텍스트, 하이퍼미디어의 기능을 지원하고, 누구나 사용할 수 있을 만큼 간단하며, 특별한 데이터 타입이 사용되지 않고 단순한 텍스트이기 때문에 웹의 발전과 일반화에 큰 공헌을 하였다. HTML 또한 SGML(Standard Generalized Markup Language)을 단순화시켜 만든 것으로, SGML을 기반으로 한 DTD(Data Type Definition)을 정의하고 그러한 정의를 따르는 웹브라우저를 사용하여 사용자가 만든 HTML인스턴스(HTML문서)를 보여주는 방식에 의한다. 그러나, HTML에서는 사용자가 임의로 태그를 만들 수 없고, 표준으로 발표된 태그 또는 각각의 브라우저에서 표현 가능한 태그만을 사용해야하는 불편함이 있다. 이와 같은 HTML의 한계를 극복하기 위해 등장한 마크업언어가 바로 XML(eXtensible Markup Language)이다.

XML은 DTD를 통해 사용자가 태그를 정의할 수 있고, 정의된 태그를 사용함으로써 문서의 「구조적인 표현」이 가능하다. 사용자 에이전트는 XML문서와 함께 전송된 DTD를 참조하여 XML문서를 처리하게 된다. 더불어, XML은 스타일시트를 사용함으로써 문서의 내용과 표현을 분리할 수 있다. 이에, 표현규칙이 담긴 파일 하나만을 변경함으로써 그 규칙을 따르는 모든 문서들의 표현방식을 일괄적으로 변경할 수 있게 된다.

발명이 이루고자 하는 기술적 과제

따라서, 본 발명의 목적은, XML을 기반으로 프로그램된 어플리케이션을 제공할 수 있는 XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체를 제공하는 것이다.

본 발명의 다른 목적은, 어플리케이션의 업데이트가 용이한 XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체를 제공하는 것이다.

본 발명의 또 다른 목적은, 보안관리가 용이한 XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체를 제공하는 것이다.

본 발명의 또 다른 목적은, 개별화된 서비스가 용이한 XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체를 제공하는 것이다.

발명의 구성 및 작용

상기 목적은, 본 발명에 따라, 복수의 컴포넌트, 및 상기 컴포넌트를 핸들링하기 위한 핸들러에 대한 등록정보 문서로서 DTD(Data Type Definition)의 문법에 따라 작성된 CTD(Component Type Definition) 문서; 및 상기 CTD문서에 등록된 컴포넌트 및 핸들러가 선언된 어플리케이션 페이지의 리스트를 포함하는 어플리케이션 정보가 저장된 어플리케이션 정보 저장부; 고객의 컴퓨터에 탑재된 컨테이너의 요청에 따라, 상기 어플리케이션 정보 저장부에 저장된 어플리케이션 정보를 제공하기 위한 어플리케이션 정보 제공 서버; 상기 어플리케이션 페이지가 저장된 어플리케이션 페이지 저장부; 상기 제공받은 어플리케이션 페이지의 리스트에 기초한 상기 컨테이너의 요청에 따라 상기 어플리케이션 페이지 저장부에 저장된 어플리케이션 페이지를 제공하기 위한 어플리케이션 페이지 제공 서버; 상기 CTD 문서에 정의된 핸들러를 실행하기 위한 핸들러 실행파일이 저장된 핸들러 실행파일 저장부; 상기 CTD 문서에 정의된 컴포넌트를 실행하기 위한 실행파일이 저장된 컴포넌트 실행파일 저장부; 및 상기 제공받은 어플리케이션 페이지를 실행하기 위한 상기 컨테이너의 요청에 따라 상기 핸들러 실행파일 저장부에 저장된 핸들러 실행파일을 제공하며, 상기 핸들러 실행파일이 실행됨에 따라 활성화된 핸들러의 요청에 따라 상기 컴포넌트 실행파일 저장부에 저장된 대응 컴포넌트 실행파일을 제공하기 위한 컴포넌트 제공 서버를 포함하는 것을 특징으로 하는 XML 기반 어플리케이션 제공 시스템에 의해 달성된다.

상기 어플리케이션 정보는, 상기 어플리케이션 페이지에 대한 권한설정정보를 더 포함하는 것이 바람직하다.

상기 컨테이너는, 상기 어플리케이션 정보 제공 서버로 상기 어플리케이션 정보를 요청하기 위한 어플리케이션 정보 요청자; 상기 어플리케이션 제공 서버로 상기 어플리케이션 페이지를 요청하기 위한 어플리케이션 페이지 로더; 및 상기 컴포넌트 제공 서버로 상기 핸들러 실행파일을 요청하고, 상기 핸들러 실행파일이 실행됨에 따라 활성화된 핸들러를 통해 상기 컴포넌트 제공 서버로 상기 컴포넌트 실행파일을 요청하기 위한 컴포넌트 로더를 포함하는 것이 바람직하다.

상기 CTD문서는, 상기 컴포넌트에 대한 정의, 상기 컴포넌트 생성, 상기 컴포넌트 간의 관계설정, 상기 컴포넌트의 속성, 및 상기 컴포넌트의 실행에 관해 정의된 태그를 포함하고, 자신의 DTD(Data Type Definition)을 구비하며, 상기 어플리케이션 페이지는, 상기 태그를 사용하여 작성된 문서로서, 상기 컨테이너는, 상기 CTD문서에 기초하여 상기 어플리케이션 페이지에 기재된 태그를 해석하고, 대응 핸들러 실행파일을 실행시켜 활성화된 핸들러를 통해 컴포넌트 실행파일을 실행시킴으로써 상기 어플리케이션 페이지가 실행되도록 하는 것이 바람직하다.

상기 어플리케이션 페이지는, 상기 컨테이너에 의해 해석되어 실행되는 객체로서의 페이지 어플리케이션의 인터페이스에 대한 정의, 상기 인터페이스에 대한 구현, 상기 CTD문서에 정의된 컴포넌트의 인스턴스 선언과 상기 컴포넌트 간의 구조적 관계, 및 이벤트처리를 포함하고, 스크립트언어로 코딩된 프로그램을 포함하는 것이 더욱 바람직하다.

한편, 본 발명의 다른 분야에 따르면, 상기 목적은, 복수의 컴포넌트, 및 상기 컴포넌트와 소정 컨테이너를 인터페이싱하기 위한 핸들러에 대한 등록정보 문서로서 DTD(Data Type Definition) 문법에 따라 작성된 CTD (Component Type Definition) 문서; 및 소정 어플리케이션을 구성하는 객체인 페이지 어플리케이션으로 구현되며 상기 CTD문서에 등록된 컴포넌트 및 핸들러가 선언된 어플리케이션 페이지의 리스트를 포함하는 어플리케이션 정보; 적어도 하나의 상기 어플리케이션 페이지; 상기 CTD 문서에 정의된 핸들러를 실행하기 위한 핸들러 실행파일; 상기 컴포넌트를 실행하기 위한 컴포넌트 실행파일; 및 상기 CTD 문서에 기초하여 적어도 하나의 상기 어플리케이션 페이지에 포함된 핸들러의 상기 핸들러 실행화일을 실행시키며, 활성화된 핸들러를 통해 대응되는 상기 컴포넌트 실행파일을 실행시킴으로써 상기 어플리케이션 페이지를 실행하기 위한 컨테이너가 기록된 것을 특징으로 하는 컴퓨터 판독가능한 기록매체에 의해서도 달성된다.

상기 어플리케이션 정보는, 상기 어플리케이션 페이지 및 컴포넌트에 대한 권한설정정보를 더 포함하는 것이 바람직하다.

상기 컨테이너는, 상기 어플리케이션 정보 제공 서버로 상기 어플리케이션 정보를 요청하기 위한 어플리케이션 정보 요청자; 상기 어플리케이션 제공 서버로 상기 어플리케이션 페이지를 요청하기 위한 어플리케이션 페이지 로더; 및 상기 컴포넌트 제공 서버로 상기 컴포넌트 실행파일을 요청하기 위한 컴포넌트 로더를 포함하는 것이 바람직하다.

상기 CTD문서는, 상기 핸들러에 대한 정의, 상기 핸들러의 생성, 상기 핸들러의 속성, 및 상기 핸들러의 실행에 관해 정의된 태그와, 상기 컴포넌트에 대한 정의, 상기 컴포넌트 생성, 상기 컴포넌트 간의 관계설정, 상기 컴포넌트의 속성, 및 상기 컴포넌트의 실행에 관해 정의된 태그, 및 스크립트언어를 지원하는 태그를 포함하고, 자신의 DTD(Data Type Definition)을 구비하며, 상기 어플리케이션 페이지는, 상기 태그를 사용하여 작성된 문서이며, 상기 컨테이너는, 상기 CTD문서에 기초하여 상기 어플리케이션 페이지에 기재된 태그를 해석하여 대응 핸들러 실행파일 또는 컴포넌트 실행파일을 실행시키며, 스크립트언어로 코딩된 프로그램을 실행함으로써 상기 어플리케이션이 실행되도록 하는 것이 바람직하다.

상기 어플리케이션 페이지는, 상기 컨테이너에 의해 해석되어 실행되는 객체로서의 페이지 어플리케이션의 인터페이스에 대한 정의, 상기 인터페이스에 대한 구현, 스크립트언어의 지원, 상기 CTD문서에 정의된 컴포넌트의 인스턴스 선언과 상기 컴포넌트 간의 구조적 관계, 및 이벤트처리가 포함되는 것이 특히 바람직하다.

이하 첨부도면을 참조하여 본 발명에 대해 상세히 설명한다.

도 1은 XML 기반 어플리케이션 제공 시스템의 제1 실시예에 따른 XML 기반 어플리케이션 제공 서버 시스템의 적용도이다.

도 1을 참조하면, XML 기반 어플리케이션 제공 서버 시스템(1)은 인터넷/인트라넷(5)을 통해 고객의 컴퓨터(3a,3n)에 XML 기반 어플리케이션을 제공한다.

XML 기반 어플리케이션 제공 서버 시스템(1)은, 본 발명의 바람직한 실시예로서 인터넷/인트라넷(5) 상에서 본 발명에 따라 XML 기반 어플리케이션을 제공하기 위한 ASP사업자의 호스트 시스템을 의미한다. XML 기반 어플리케이션은 본 발명에 따라 XML문법에 근거하여 코딩된 어플리케이션으로, 소프트웨어 공학적인 측면에서 컴포넌트 기반 구조를 가진다. 객체지향 프로그래밍(OOP; Object-Oriented Programming)에서, 컴포넌트는 재사용이 가능한 프로그램 빌딩블록으로서, 소정 어플리케이션을 형성하기 위해 같은 컴퓨터에 있거나 또는 네트웍 상의 다른 컴퓨터에 있는 컴포넌트들과 조합될 수 있으며 필요한 서비스를 위해 서로 통신할 수 있다. 컴포넌트는 컨테이너라 불리는 일종의 소프트웨어 내에서 동작한다. 대표적인 컨테이너로는 EJB(Enterprize Java Beans) 컨테이너를 들 수 있다. 본 발명에서는 사용자 에이전트가 컨테이너 역할을 한다.

본 발명에 따른 XML 기반 어플리케이션에 대해 보다 구체적으로 설명하면 다음과 같다.

도 2는 본 발명에 따른 XML 기반 어플리케이션의 구조를 설명하기 위한 참고도이다.

도 2를 참조하면, XML 기반 어플리케이션은, 컴포넌트와 핸들러를 포함한다. 여기서, 핸들러는 일종의 컴포넌트로서, 컨테이너와 컴포넌트를 인터페이싱해준다. 이와 같은 구조에 의해 XML 기반 어플리케이션은 종래 개발되어 사용되고 있는 컴포넌트를 그대로 사용할 수 있게 된다. XML 기반 어플리케이션의 소스에 실제로 표현된 컴포넌트와 핸들러의 예는 다음과 같다. 여기서, 컴포넌트는 「JFrame」이고, 「JFrame」을 핸들링하는 핸들러로는 「SwingHandler」가 선언되어 있다.

```
<JFrame x="200" y="200" width="600" height="500"
handler="SwingHandler"/>
```

CTD문서는 DTD를 바탕으로 작성된 XML 기반 어플리케이션의 스키마이다. XML 기반 어플리케이션은, XML의 문법을 사용하므로, 개발자가 정의한 태그의 집합 및 그 의미를 정의한 DTD가 존재하는 것은 당연하다. 다만, DTD는 확장된 배커스-노어 폼(Extended Backus-Naur Form, EBNF)라는 수정된 표기방식을 사용하는 바, 그 표기방식이 단순하여 기술(description) 능력이 제한되어 있고 확장성이 없는 단점이 있다. 이에, XML을 바탕으로 XML이 사용하는 스키마를 기술한다는 개념이 도입되었으며, 제안된 것으로는 XML-Data, DCD(Document Content Description for XML, DCD)가 있다. 즉, 본 발명에서의 CTD문서는 스키마가 기술된 문서이다.

보다 구체적으로, CTD(Component Type Definition) 문서는 당해 XML 기반 어플리케이션에서 사용될 컴포넌트와 그 핸들러에 대한 등록정보를 포함한다. 즉, CTD문서에는 XML 기반 어플리케이션의 소스에 엘리먼트로서 포함될 컴포넌트에 대한 정의 및 정의된 컴포넌트를 핸들링할 핸들러에 대한 선언이 포함된다. 컴포넌트에 대한 정의는 다시 컴포넌트의 생성, 컴포넌트 간의 관계설정, 컴포넌트의 실행에 관한 세부적인 정의로 나뉘어진다.

CTD문서에 컴포넌트의 등록은 엘리먼트의 하나인 <ElementType>에 의해 이루어진다. <ElementType>은 그 속성(Atrribute)으로서 id, class, handler, handlerClass를 가질 수 있다.

컴포넌트의 하나인 「JFrame」에 대한 정의는 다음과 같다.

```
<ElementType id="JFrame" class="Javax.swing.JFrame"
handler="Swinghandler">
</ElementType>
```

여기서, Id는 컴포넌트를 나타내는 엘리먼트 태그가 되며, 그 예는 다음과 같다.

```
<JFrame>
</JFrame>
```

「class」는 컴포넌트의 실제 이름으로서 대응 핸들러가 컴포넌트를 활성화하고자 할 때 필요하다. 즉, 「class」는 Java에서의 클래스 명칭에 해당되는 것이다.

「handler」는 컴포넌트를 핸들링하는 핸들러를 가리키는 바, JFrame을 핸들링하는 핸들러는 「SwingHandler」임이 정의되어 있다. 이에 따라, JFrame은 SwingHandler에 의해 활성화되게 된다. 즉, 컨테이너와 컴포넌트를 핸들러가 인터페이싱하게 되므로, 컨테이너가 XML 기반 어플리케이션 소스를 해석함에 있어 JFrame이라는 엘리먼트를 만나게 되면 CTD문서를 참조하여 JFrame의 핸들러가 SwingHandler임을 알아낸다. 이에 따라 컨테이너는 SwingHandler를 활성화하고, 활성화된 swingHandler에 의해 JFrame이 활성화되게 된다.

대안적으로, 다음과 같이, CTD문서에서 엘리먼트로서 handler를 정의하고 그 아래에 여러 ElementType을 정의할 수 있다.

```
<Handler id="SwingHandler" class="SwingHandler01">
<ElementType id="JFrame" class="Javax.swing.JFrame"/>
<ElementType id="JPanel" class="Javax.swing.JPanel"/>
<ElementType id="JTree" class="Javax.swing.JTree"/>
</Handler>
```

여기서, 컴포넌트인 「JFrame」, 「JPanel」, 「JTree」는 모두 동일한 핸들러인 SwingHandler에 의해 활성화됨을 알 수 있다.

다음으로, 컴포넌트 생성에 관한 등록정보는 <Constructor>태그를 통해 다음과 같이 표시된다.

```
<ElementType Id="Color" class="Java.awt.Color">
<Constructor>
∘∘<Param  class="int" value="[red]" />
∘∘<Param  class="int" value="[green]" />
∘∘<Param  class="int" value="[blue]" />
</Constructor>
</ElementType>
```

컴포넌트 Color의 핸들러는 Color를 생성할 때 Constructor의 등록정보를 참조하는 바, 위의 경우는 Color의 생성자 중에서 Color(int arg1,Int arg2,Int arg3)형의 생성자를 기동할 것을 의미한다. 또한, 각 값은 속성(Attribute)에서 입력받을 것을 지시하고 있다.

이와 같은 등록정보 하에, 실제 XML 기반 어플리케이션 소스에서 Color는 다음과 같이 표시될 수 있으며, 이에 핸들러는 Color::Color(100,0,0)이라는 생성자로 Color 컴포넌트를 생성하게 된다.

```
<Color red="100" green="0" blue="0"/>
```

다음으로, 컴포넌트 간의 관계설정에 관한 등록정보는 <BuildRelation>을 통해 다음과 같이 표시된다.

```
<ElementType Id="Color" class="Java.awt.Color">
<Constructor>
<Param   class="int" value="[red]"  />
<Param   class="int" value="[green]"  />
<Param   class="int" value="[blue]"  />
</Constructor>
<BuildRelation>
∘<Parents>
∘<Parent class="Javax.swing.JComponent">
∘∘<Method name="setBackground">
∘∘<Param class="Java.awt.Color" value="[this]" />
∘∘</Method>
∘∘</Parent>
∘</Parents>
</BuildRelation>
</ElementType>
```

Color 컴포넌트의 핸들러는 Color를 생성한 다음 이를 다른 컴포넌트와 관계를 형성시키는 바, 이때 참조될 정보가 <BuildRelation>이라는 엘리먼트에 의해 정의된다. XML에서와 마찬가지로, Parents 엘리먼트의 하위에는 여러 개의 Parent가 올 수 있다.

이와 같은 등록정보 하에, 실제 XML 기반 어플리케이션 소스에서 관계는 다음과 같이 표시될 수 있다.

```
<JTextField>
<Color red="100" green="0" blue="0"/>
</JTextField>
```

여기서, 「Color」는 「JTextfield」라는 Parent 컴포넌트에 의해 소정 메소드에 의해 호출될 수 있도록 관계맺고 있음을 알 수 있다.

다음으로, 컴포넌트의 속성에 관한 등록정보는 <Attribute>을 통해 다음과 같이 표시된다.

```
<ElementType Id="JLable" class="Javax.swing.JLabel">
∘<Attributes>
∘∘<Attribute Id="text"  class="Java.lang.String">
∘∘<Method name="setText">
∘∘<Param class="Java.lang.String" value="[this]"/>
∘∘∘∘</Method>
∘      </Attribute>
∘</Attributes>
</ElementType>
```

여기서, 「JLabel」 컴포넌트는 「text」라는 속성을 가지게 되며, 그 속성의 값은 「String」형임을 알 수 있다. String값은 JLabel의 소정 메소드를 통해 설정되게 된다.

다음으로, 컴포넌트의 실행에 관한 등록정보는 <Run>을 통해 다음과 같이 표시된다.

```
<ElementType Id="JFrame" class="Javax.swing.JFrame">
∘<Run>
```

```
    <Method name="show">
    </Method>
  </Run>
</ElementType>
```

위의 경우에, 「JFrame」 컴포넌트가 생성되고, 속성이 할당된 후 실행되게 되는데 이때 기동되는 메소드는 「show」가 된다.

전술한 바와 같이, CTD문서는 아래의 CTD문서에 대한 DTD에 기초하여 작성된다. 이에 기초하여 XML 문법에 따라 적어도 하나의 어플리케이션 페이지가 작성되며, 이들 어플리케이션 페이지가 조합되어 하나의 어플리케이션 소스를 이루게 되는 것이다.

```
  <!이것은 CTD의 DTD입니다.>
<!이것은 CTD의 DTD입니다.>
<!ELEMENT AMLTagHandlers (Handler)*>
<!ELEMENT Handler (elementType)*>
<!ATTLIST Handler id CDATA #REQUIRED>
<!ATTLIST Handler class CDATA #REQUIRED>
<!ELEMENT elementType  (Constructor?,BuildRelation?,Attributes?,Run?) >
<!ATTLIST elementType id CDATA #REQUIRED>
<!ATTLIST elementType class CDATA #REQUIRED>
<!ATTLIST elementType handler CDATA #IMPLIED>
<!ATTLIST elementType handler_class CDATA #IMPLIED>
<!ELEMENT Constructor (Param*) >
<!ELEMENT BuildRelation (Parents?) >
<!ELEMENT Parents (Parent*)>
<!ELEMENT Parent (Method+)>
<!ATTLIST Parent class CDATA #REQUIRED>
<!ELEMENT Attributes (Attribute*)>
<!ELEMENT Attribute (Method)>
<!ATTLIST Attribute id CDATA #REQUIRED>
<!ATTLIST Attribute class CDATA #REQUIRED>
<!ELEMENT Run (Method)>
<!ELEMENT Method (Param*)>
<!ATTLIST Method name CDATA #REQUIRED>
<!ELEMENT Param EMPTY>
<!ATTLIST Param class CDATA #REQUIRED>
<!ATTLIST Param value CDATA #REQUIRED>
```

컨테이너(사용자 에이전트)는 전술한 CTD문서를 참조하여 어플리케이션 페이지를 해석하여 실행한다. 어플리케이션 페이지가 실행되어 해석중인 객체를 페이지 어플리케이션이라고 부른다. 페이지 어플리케이션은 조합되어 하나의 완전한 어플리케이션을 이루게 된다. 페이지 어플리케이션은 자신의 속성과 메소드를 가지는 객체이며 프로그램모듈로서 행동한다.

따라서, 페이지 어플리케이션의 인터페이스에 대한 정의는 <Interface>에 의해 다음과 같이 표시된다.

```
<?xml version="1.0" encoding="EUC-KR" ?>
<aml version=".10">
<Interface>
 <Attribute type="String" name="jjj"/>
 <Method name="go">
  <Param name="where" type="String"/>
 </Method>
</Interface>
</aml>
```

여기서, <aml>은 application markup language의 머릿글자를 따서 만든 조어 「aml」을 따서 만든 루트 엘리먼트 태그이다.

페이지 어플리케이션의 인터페이스에 대한 실행은 <Implement>에 의해 다음과 같이 표시된다.

```
<?xml version="1.0" encoding ="EUC-KR" ?>
<aml version=".10">
<Interface name="aPage">
 <Attribute type="String" name="jjj"/>
 <Method name="update">
  <Param name="where" type="String"/>
 </Method>
</Interface>
<Implement interface="aPage">
```

```
 <AttributeImple name="jjj">
<Set>
<!Some code
</Set>
<Get>
<!Some code
</Get>
</AttributeImple>
<MethodImple name=update>
</MethodImple>
</Implement>
</aml>
```

페이지 어플리케이션의 스크립트 언어 지원은 <Script>에 의해 다음과 같이 표시된다.

```
<Implement interface="aPage">
<MethodImple name="update">
<Script language="aml">(이하 계속)
```

```
 <Call-method id="tmp" target="ddd" name="go">
  <ReferenceOf target="ttt"/>
 </Call-method>
</Script>
</MethodImple>
</Implement>
```

스크립트 언어는 필요에 따라 실행가능한 어떤 부분에도 삽입될 수 있는 바, 주로 페이지 어플리케이션의 통신을 위해 사용된다.

다음은 XML 기반 어플리케이션으로서 문서편집기의 CTD문서의 일 예는 다음과 같다.

```
<?xml version=1.0 encoding=EUC-KR ?>
<!--Default Handler Configuration -->
<AMLTagHandlers>
<!-- AML Master Component -->
<Handler id=BaseHandler class=ABXCompHandlerBase >
•<elementType id=aml class=undefined/>
•<elementType id="Components" class="undefined"/>
•<elementType id="Startup" class="undefined"/>
•<elementType id="EventHandlers" class="undefined"/>
</Handler>
<!-- AML InstanceOf Handler -->
<Handler id="InstanceOfHandler" class="ABXInstanceOfHandler" >
•<elementType id="InstanceOf" class="undefined"/>
</Handler>
<!-- AML Swing Menu Handler -->
<Handler id="SwingMenuHandler" class="ABXSwingMenuHandler">
•<elementType id="JMenuItem" class="javax.swing.JMenuItem"/>
•<elementType id="JMenu" class="javax.swing.JMenu"/>
•<elementType id="JMenuBar" class="javax.swing.JMenuBar"/>
</Handler>
<!-- AML Swing ActionEvent Handler -->
<Handler id="SwingEventHandler" class="ABXEventHandler" >
•<elementType id="Action" class="ABXActionListener"/>
•<elementType id="Key" class="ABXKeyListener"/>
</Handler>
<!-- Swing Components -->
<Handler id="SwingHandler" class="ABXSwingHandler" >
•<elementType id="JFrame" class="javax.swing.JFrame"/>
•<elementType id="JInternalFrame" class="javax.swing.JInternalFrame"/>
•<elementType id="JDesktopIcon" class="javax.swing.JInternalFrame.JDesktopIcon"/>
•<elementType id="JPanel" class="javax.swing.JPanel"/>
</Handler>(이하 계속)
```

```
<!-- 이하는 사용자가 정의한 클래스 핸들링 방법이다.
이는 AnyHandler가 담당한다.-->
```

```
<Handler id="AnyHandler" class="ABXSwingAnyHandler">
<!--Javax.swing.border.BevelBorder-->
∘<elementType id="BevelBorder" class="javax.swing.border.BevelBorder">
∘<Constructor>
∘∘<Param class="int" value="[type]"/>
∘</Constructor>
∘∘<BuildRelation>
∘∘<Parents>
∘∘∘<Parent class="javax.swing.JComponent">
∘∘∘<Method name="setBorder">
∘∘<Param class="javax.swing.border.Border" value="[this]" />
∘∘∘∘∘</Method>
∘∘∘∘</Parent>
∘∘∘</Parents>
∘∘</BuildRelation>
∘</elementType>
</Handler>
<Handler id="LayoutHandler" class="ABXLayoutHandler" >
∘<elementType id="Layout" class="undefined"/>
</Handler>
</AMLTagHandler>
```

위의 CTD문서에 기초하여 구현된 문서편집기 어플리케이션의 소스코드는 다음과 같다.

```
<?xml version="1.0" encoding="EUC-KR" ?>
<aml version=1.0>
<Components>
  <JFrame x="200" y="200" width="600" height="500" >


   <Layout type="Border" rows="3" columns="1" />
   <JMenuBar>
   ∘ <JMenu text="File">
   ∘ ∘<JMenuItem text="New"/>
∘ ∘<JMenuItem text="Open"/>
∘∘<JMenuItem text="Save" />
∘∘<JMenuItem text="Save as"/>
∘ </JMenu>
∘ <JMenu text="Edit"/>
∘ <JMenu text="View" />
   </JMenuBar>
   <JDesktopPane constraints="Center">
   ∘<BevelBorder type="1"/>
   </JDesktopPane>
   <JPanel constraints="South"/>
  </JFrame>
</Components>
<Startup>
</Startup> (이하 계속)
```

```
<!-- Event Handlers Block -->
<EventHandlers>
∘<Action name="JButton4">
∘∘<Call target="System" method="exit" >
∘∘∘<Param name="jjj">0</Param>
∘∘</Call>
∘</Action>
<EventHandlers>
</aml>
```

이하에서는 전술한 XML 기반 어플리케이션을 제공하기 위한 도 1의 XML 기반 어플리케이션 제공 서버 시스템(1) 및 그 XML 기반 어플리케이션 제공방법을 설명한다.

도 3은 도 1의 고객 컴퓨터(3a,3n) 및 XML 기반 어플리케이션 제공 서버 시스템(1)의 블록도이다.

도 3을 참조하면, XML 기반 어플리케이션 제공 서버 시스템(1)은, 어플리케이션 정보 데이터베이스(15), 어플리케이션 페이지 데이터베이스(16), 핸들러 실행파일 데이터베이스(17), 컴포넌트 실행파일 데이터베이스(18), 및 서버부(10)를 포함한다.

서버부(10)는, 어플리케이션 정보 제공 서버(11), 어플리케이션 페이지 제공 서버(12), 및 컴포넌트 제공 서버(13)를 포함한다.

어플리케이션 정보 데이터베이스(15)는, 어플리케이션 저장부로서, 어플리케이션에 대한 어플리케이션 정보가 저장되어 있다. 「어플리케이션 정보」는, CTD 문서 및 어플리케이션 페이지의 리스트를 포함한다.

어플리케이션 페이지 데이터베이스(16)는 어플리케이션 페이지 저장부로서, 어플리케이션 페이지가 저장되어 있다.

핸들러 실행파일 데이터베이스(17)는 핸들러 실행파일 저장부로서, CTD 문서에 정의된 핸들러를 활성화시키기 위한 실행파일이 저장되어 있다.

컴포넌트 실행파일 데이터베이스(18)는 컴포넌트 실행파일 저장부로서, 핸들러에 의해 CTD 문서에 정의된 컴포넌트를 활성화시키기 위한 실행파일이 저장되어 있다.

어플리케이션 정보 제공 서버(11)는 어플리케이션 정보 데이터베이스(15)에 저장된 어플리케이션 정보를 추출하여 인터넷/인트라넷(5)을 통해 고객 컴퓨터(3a,3n)로 전송한다. 어플리케이션 페이지 제공 서버(12)는 어플리케이션 페이지 데이터베이스(16)에 저장되어 있는 어플리케이션 페이지를 인터넷/인트라넷(5)을 통해 고객 컴퓨터(3a,3n)로 전송한다.

컴포넌트 제공 서버(13)는 핸들러 실행파일 데이터베이스(17) 또는 컴포넌트 실행파일 데이터베이스(18)에 저장된 핸들러 실행파일 또는 컴포넌트 실행파일을 고객 컴퓨터(3a,3n)로 전송한다.

고객의 컴퓨터(3a,3n)에는 본 발명에 따른 XML 기반 어플리케이션 소스(이하 「AML 소스」라 함)를 해석하고 처리하기 위한 유저 에이전트인 컨테이너가 탑재되어 있다. 컨테이너(30)는 어플리케이션 정보 요청자(31), 어플리케이션 페이지 로더(32), 및 컴포넌트 로더(33)를 구비한다.

어플리케이션 정보 요청자(31)는 어플리케이션 정보 제공 서버(11)로 어플리케이션 정보를 요청하며, 어플리케이션 페이지 로더(32)는 제공받은 어플리케이션 정보에 기초하여 어플리케이션 페이지 제공 서버(12)로 필요한 어플리케이션 페이지를 요청한다. 컴포넌트 로더(33)는 제공받은 어플리케이션 페이지를 해석하여 필요한 핸들러 또는 컴포넌트 실행파일을 컴포넌트 제공 서버(13)로 요청한다.

도 4는 본 발명의 제2 실시예에 따른 XML 기반 어플리케이션이 기록된 기록매체가 탑재된 컴퓨터(43)의 개략도이다.

도 4를 참조하면, 컴퓨터(43)는, 하드디스크 및 메모리를 탑재하고 상기 하드디스크 및 메모리에 연결되어 소정 프로그램을 실행하기 위한 프로세서를 구비한 컴퓨터 본체(100), 디스플레이 장치(101), 키보드(102), 및 마우스(103)를 구비한다.

상기 하드디스크에는 XML 기반 어플리케이션을 실행하기 위해 필요한 어플리케이션 정보, 어플리케이션 페이지, 핸들러 실행파일, 및 컴포넌트 실행파일이 저장되어 있으며, XML 기반 어플리케이션을 해석하기 위한 사용자 에이전트인 컨테이너는 상기 메모리에 탑재되어 상기 프로세서에 의해 실행되게 된다.

도 5는 본 발명의 XML 기반 어플리케이션 제공 시스템의 제 3실시예로서, XML 기반 어플리케이션을 실행하기 위한 적어도 하나의 컴퓨터(53a, 53n)가 연결된 네트워크 구성 도이다.

도 5를 참조하면, 컴퓨터(53a,53n)는 본 발명에 따른 XML 기반 어플리케이션을 실행하기 위해 LAN/WAN(50)에 접속되어 있다. 컴퓨터(53a,53n)는 도 4의 그것과 동일한 구조를 지니는 바, 도시않은 컴퓨터 본체에 XML 기반 어플리케이션을 실행하기 위한 컨테이너가 탑재되어 있다. 다만, XML 기반 어플리케이션을 실행하기 위해 필요한 어플리케이션 정보, 어플리케이션 페이지, 핸들러 실행파일, 및 컴포넌트 실행파일은 thin-client를 구현하기 위해 LAN/WAN(50)에 물려 있는 XML 기반 어플리케이션 제공 데이터베이스(1000)에 저장되어 있다.

도 6은 도 4의 컴퓨터(43), 또는 도 5의 컴퓨터(53a,53n) 및 XML 기반 어플리케이션 제공 데이터베이스(1000))의 블록도이다.

도 6을 참조하면, XML 기반 어플리케이션 제공 데이터베이스(1000) 또는 컴퓨터(43)의 본체에는, 어플리케이션 정보 파일(150), 어플리케이션 페이지 파일(160), 핸들러 실행파일(170), 및 컴포넌트 실행파일(180)이 저장되어 있다.

어플리케이션 정보 파일(150)은, 「어플리케이션 정보」로서, CTD 문서 및 어플리케이션 페이지의 리스트가 포함되어 있다. 어플리케이션 페이지 파일(160)은 어플리케이션 페이지가 포함되어 있으며, 핸들러 실행파일(170)은 CTD 문서에 정의된 핸들러를 활성화시키기 위한 실행파일이며, 컴포넌트 실행파일(180)은 대응 핸들러에 의해 컴포넌트를 활성화시키기 위한 실행파일이다.

한편, 컴퓨터(43,53a,53n)의 도시않은 메모리에는 AML 소스를 해석하기 위한 컨테이너(300)가 탑재되어 실행되는 바, 컨테이너(300)는 어플리케이션 정보 요청자(310), 어플리케이션 페이지 로더(320), 및 컴포넌트 로더(330)를 구비한다.

어플리케이션 정보 요청자(310)는 어플리케이션 정보 실행파일(150)을 가져오고, 어플리케이션 페이지 로더(320)는 가져온 어플리케이션 정보에 기초하여 어플리케이션 페이지 파일(160)로부터 필요한 어플리케이션 페이지를 가져온다. 컴포넌트 로더(330)는 가져온 어플리케이션 페이지를 해석하여 필요한 핸들러 또는 컴포넌트 실행파일을 가져온다.

상기와 같은 구성에 따라 본 발명에 따른 XML 기반 어플리케이션 제공방법을 설명하면 다음과 같다.

도 7은 어플리케이션 제공방법에 의해 어플리케이션이 실행되는 과정을 설명하기 위한 참고도이다. 다만, 사용자는 도 1, 4, 또는 5의 컴퓨터를 사용하는 자를 가리키며, 컨테이너는 도 1, 4, 또는 5의 컴퓨터에 탑재되어 실행되는 사용자 에이전트로서 컨테이너를 가리킨다. 파일저장 시스템은, 도 1의 XML 기반 어플리케이션 제공 서버 시스템(1), 도 4의 컴퓨터(43)의 본체 또는 도 5의 XML 기반 어플리케이션 제공 데이터베이스(1000)를 가리킨다.

도 7에 도시된 바와 같이, 사용자가 어플리케이션을 요구하면(①단계), 즉 컴퓨터를 사용하여 소정 어플리케이션을 실행하고자 하면, 컨테이너는 파일저장 시스템에게 어플리케이션 정보를 요구한다(②단계).

이에, 컨테이너는 파일저장 시스템으로부터 대응 어플리케이션 정보로서 CTD 문서, 어플리케이션 페이지 리스트, 및 권한설정정보를 수신한다(③단계). 여기서, 권한설정정보는 각 어플리케이션 페이지 및 컴포넌트에 대한 접근권한을 설정한 것이다. 접근권한 설정은 소정 운영체계를 기반으로 다양한 네트워크 접속 어플리케이션을 통해 지원되고 있다. 이에 의해, 소정 문서에 대한 접근권한을 설정하는 방법을 통해 소정 페이지 어플리케이션에 의한 서비스 제공을 제한하거나 허여할 수 있게 된다.

어플리케이션 정보를 수신받은 컨테이너는 리스트에 기재된 순서에 따라 어플리케이션 페이지를 해석하고 해석된 어플리케이션 페이지에 포함된 핸들러를 요구한다(④단계).

이에, 파일저장 시스템은 핸들러 실행파일을 제공하고(⑤단계), 컨테이너는 실행파일을 실행시켜 핸들러를 활성화시킨다.

활성화된 핸들러는 컨테이너에 의해 지시된 적어도 하나의 대응 컴포넌트를 요구하고(⑥단계), 파일저장 시스템은 요구된 컴포넌트 실행파일을 제공한다(⑦단계)

핸들러에 의해 활성화된 적어도 하나의 컴포넌트에 의해 페이지 어플리케이션이 생성된다(⑧단계).

생성된 페이지 어플리케이션은 사용자에게 소정 서비스를 제공하고(⑨단계), 사용자는 필요한 경우 페이지 어플리케이션 간에 또는 페이지 어플리케이션 내에서 이벤트를 발생시킨다(⑩단계).

발생된 이벤트는 페이지 어플리케이션에 의해 처리된다(⑪단계).

생성된 페이지 어플리케이션과 다른 페이지 어플리케이션 간에 발생된 이벤트인 경우 필요한 어플리케이션 페이지에 정의된 핸들러를 요구하고(④단계) 이하 단계가 반복되어 이벤트에 관련된 페이지 어플리케이션이 생성된다.

현재 생성된 페이지 어플리케이션 내에서 발생된 이벤트인 경우, 내부 컴포넌트에 의해 응답처리된다.

한편, 어플리케이션 페이지에는 필요에 따라 복수개의 핸들러를 포함하는 것이 일반적이므로, 상기 ④ 내지 ⑦단계는 복수회 반복되어 실행될 수 있다.

발명의 효과

이상 설명한 바와 같이, 본 발명에 따르면, XML을 기반으로 프로그램된 어플리케이션을 제공할 수 있는 XML 기반 어플리케이션 제공 시스템, 및 XML 기반 어플리케이션이 기록된 컴퓨터 판독가능한 기록매체가 제공된다.

특히, ASP사업자의 경우, 컨테이너만을 고객의 컴퓨터에 설치하고, 나머지 필요한 파일은 서버 측에 저장시켜 둠으로써, 어플리케이션의 업데이트가 필요한 경우 제공되는 파일만을 변경하거나 수정하는 방식에 의해 용이하게 업데이트할 수 있다. 더불어, 전술한 바와 같이 어플리케이션 페이지에 접근권한을 설정함으로써 용이하고 강력한 보안관리를 할 수 있다. 나아가, 스타일시트를 사용함으로써 한 벌의 소스만으로도 고객마다 개별화된 페이지 어플리케이션을 제공할 수 있게 된다.

(57) 청구의 범위

청구항 1.

복수의 컴포넌트, 및 상기 컴포넌트를 핸들링하기 위한 핸들러에 대한 등록정보 문서로서 DTD(Data Type Definition)의 문법에 따라 작성된 CTD(Component Type Definition) 문서; 및 상기 CTD문서에 등록된 컴포넌트 및 핸들러가 선언된 어플리케이션 페이지의 리스트를 포함하는 어플리케이션 정보가 저장된 어플리케이션 정보 저장부;

고객의 컴퓨터에 탑재된 컨테이너의 요청에 따라, 상기 어플리케이션 정보 저장부에 저장된 어플리케이션 정보를 제공하기 위한 어플리케이션 정보 제공 서버;

상기 어플리케이션 페이지가 저장된 어플리케이션 페이지 저장부;

상기 제공받은 어플리케이션 페이지의 리스트에 기초한 상기 컨테이너의 요청에 따라 상기 어플리케이션 페이지 저장부에 저장된 어플리케이션 페이지를 제공하기 위한 어플리케이션 페이지 제공 서버;

상기 CTD 문서에 정의된 핸들러를 실행하기 위한 핸들러 실행파일이 저장된 핸들러 실행파일 저장부;

상기 CTD 문서에 정의된 컴포넌트를 실행하기 위한 실행파일이 저장된 컴포넌트 실행파일 저장부; 및

상기 제공받은 어플리케이션 페이지를 실행하기 위한 상기 컨테이너의 요청에 따라 상기 핸들러 실행파일 저장부에 저장된 핸들러 실행파일을 제공하며, 상기 핸들러 실행파일이 실행됨에 따라 활성화된 핸들러의 요청에 따라 상기 컴포넌트 실행파일 저장부에 저장된 대응 컴포넌트 실행파일을 제공하기 위한 컴포넌트 제공 서버를 포함하는 것을 특징으로 하는 XML 기반 어플리케이션 제공 시스템.

청구항 2.

제1항에 있어서,

상기 어플리케이션 정보는, 상기 어플리케이션 페이지에 대한 권한설정정보를 더 포함하는 것을 특징으로 하는 XML 기반 어플리케이션 제공 시스템.

청구항 3.

제1항 또는 제2항에 있어서,

제3항에 있어서,

상기 컨테이너는,

상기 어플리케이션 정보 제공 서버로 상기 어플리케이션 정보를 요청하기 위한 어플리케이션 정보 요청자;

상기 어플리케이션 제공 서버로 상기 어플리케이션 페이지를 요청하기 위한 어플리케이션 페이지 로더; 및

상기 컴포넌트 제공 서버로 상기 핸들러 실행파일을 요청하고, 상기 핸들러 실행파일이 실행됨에 따라 활성화된 핸들러를 통해 상기 컴포넌트 제공 서버로 상기 컴포넌트 실행파일을 요청하기 위한 컴포넌트 로더를 포함하는 것을 특징으로 하는 XML 기반 어플리케이션 제공 시스템.

청구항 4.
제3항에 있어서,

상기 CTD문서는, 상기 컴포넌트에 대한 정의, 상기 컴포넌트 생성, 상기 컴포넌트 간의 관계설정, 상기 컴포넌트의 속성, 및 상기 컴포넌트의 실행에 관해 정의된 태그를 포함하고, 자신의 DTD(Data Type Definition)을 구비하며,

상기 어플리케이션 페이지는, 상기 태그를 사용하여 작성된 문서로서,

상기 컨테이너는, 상기 CTD문서에 기초하여 상기 어플리케이션 페이지에 기재된 태그를 해석하고, 대응 핸들러 실행파일을 실행시켜 활성화된 핸들러를 통해 컴포넌트 실행파일을 실행시킴으로써 상기 어플리케이션 페이지가 실행되도록 하는 것을 특징으로 하는 XML 기반 어플리케이션 제공 시스템.

청구항 5.
제4항에 있어서,

상기 어플리케이션 페이지는, 상기 컨테이너에 의해 해석되어 실행되는 객체로서의 페이지 어플리케이션의 인터페이스에 대한 정의, 상기 인터페이스에 대한 구현, 상기 CTD문서에 정의된 컴포넌트의 인스턴스 선언과 상기 컴포넌트 간의 구조적 관계, 및 이벤트처리를 포함하고, 스크립트언어로 코딩된 프로그램을 포함하는 것을 특징으로 하는 XML 기반 어플리케이션 제공 시스템.

청구항 6.
복수의 컴포넌트, 및 상기 컴포넌트와 소정 컨테이너를 인터페이싱하기 위한 핸들러에 대한 등록정보 문서로서 DTD(Data Type Definition) 문법에 따라 작성된 CTD (Component Type Definition) 문서; 및 소정 어플리케이션을 구성하는 객체인 페이지 어플리케이션으로 구현되며 상기 CTD문서에 등록된 컴포넌트 및 핸들러가 선언된 어플리케이션 페이지의 리스트를 포함하는 어플리케이션 정보;

적어도 하나의 상기 어플리케이션 페이지;

상기 CTD 문서에 정의된 핸들러를 실행하기 위한 핸들러 실행파일;

상기 컴포넌트를 실행하기 위한 컴포넌트 실행파일; 및

상기 CTD 문서에 기초하여 적어도 하나의 상기 어플리케이션 페이지에 포함된 핸들러의 상기 핸들러 실행화일을 실행시키며, 활성화된 핸들러를 통해 대응되는 상기 컴포넌트 실행파일을 실행시킴으로써 상기 어플리케이션 페이지를 실행하기 위한 컨테이너가 기록된 것을 특징으로 하는 컴퓨터 판독가능한 기록매체.

청구항 7.
제6항에 있어서,

상기 어플리케이션 정보는, 상기 어플리케이션 페이지 및 컴포넌트에 대한 권한설정정보를 더 포함하는 것을 특징으로 하는 컴퓨터 판독가능한 기록매체.

청구항 8.
제6항 또는 제7항에 있어서,

상기 컨테이너는,

상기 어플리케이션 정보 제공 서버로 상기 어플리케이션 정보를 요청하기 위한 어플리케이션 정보 요청자;

상기 어플리케이션 제공 서버로 상기 어플리케이션 페이지를 요청하기 위한 어플리케이션 페이지 로더; 및

상기 컴포넌트 제공 서버로 상기 컴포넌트 실행파일을 요청하기 위한 컴포넌트 로더를 포함하는 것을 특징으로 하는 컴퓨터 판독가능한 기록매체.

청구항 9.
제8항에 있어서,

상기 CTD문서는, 상기 핸들러에 대한 정의, 상기 핸들러의 생성, 상기 핸들러의 속성, 및 상기 핸들러의 실행에 관해 정의된 태그와, 상기 컴포넌트에 대한 정의, 상기 컴포넌트 생성, 상기 컴포넌트 간의 관계설정, 상기 컴포넌트의 속성, 및 상기 컴포넌트의 실행에 관해 정의된 태그, 및 스크립트언어를 지원하는 태그를 포함하고, 자신의 DTD(Data Type Definition)을 구비하며,

상기 어플리케이션 페이지는, 상기 태그를 사용하여 작성된 문서이며,

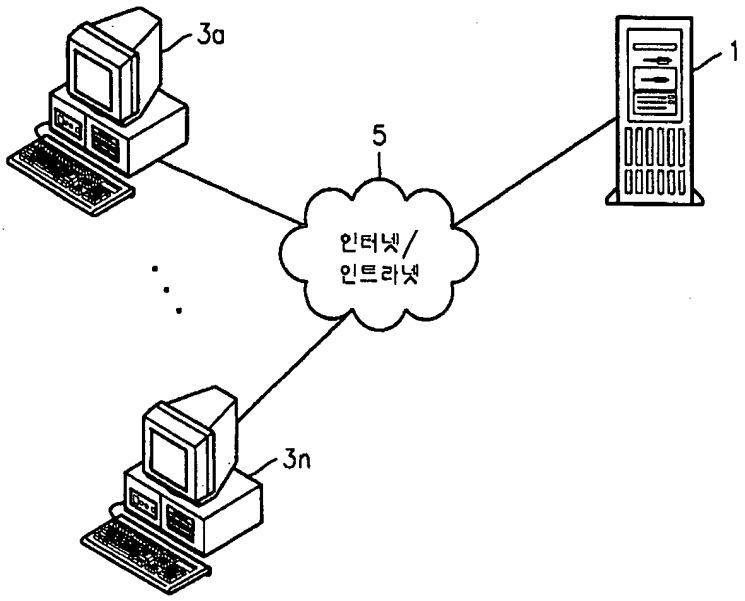
상기 컨테이너는, 상기 CTD문서에 기초하여 상기 어플리케이션 페이지에 기재된 태그를 해석하여 대응 핸들러 실행파일 또는 컴포넌트 실행파일을 실행시키며, 스크립트언어로 코딩된 프로그램을 실행함으로써 상기 어플리케이션이 실행되도록 하는 것을 특징으로 하는 컴퓨터 판독가능한 기록매체.

청구항 10.
제9항에 있어서,

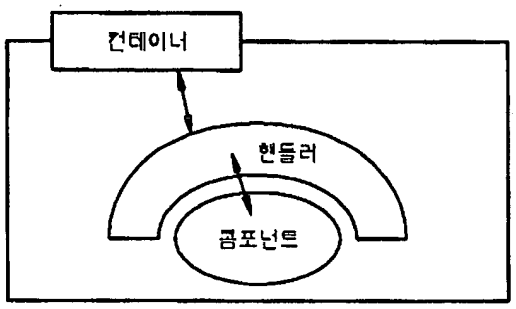
상기 어플리케이션 페이지는, 상기 컨테이너에 의해 해석되어 실행되는 객체로서의 페이지 어플리케이션의 인터페이스에 대한 정의, 상기 인터페이스에 대한 구현, 스크립트언어의 지원, 상기 CTD문서에 정의된 컴포넌트의 인스턴스 선언과 상기 컴포넌트 간의 구조적 관계, 및 이벤트처리가 포함되는 것을 특징으로 하는 컴퓨터 판독가능한 기록매체.

도면

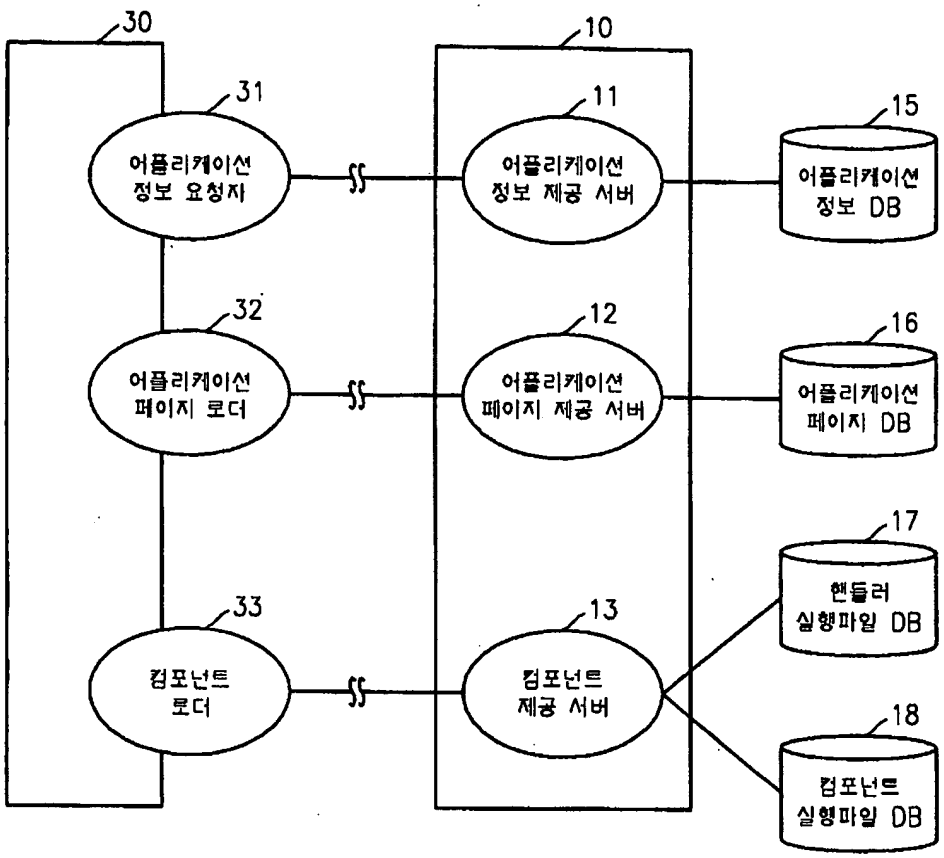
도면 1



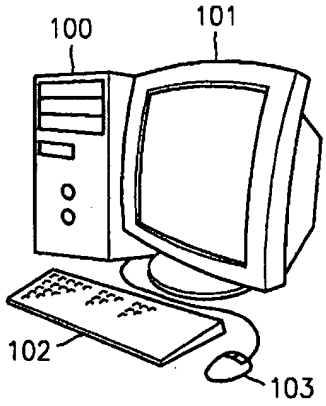
도면 2



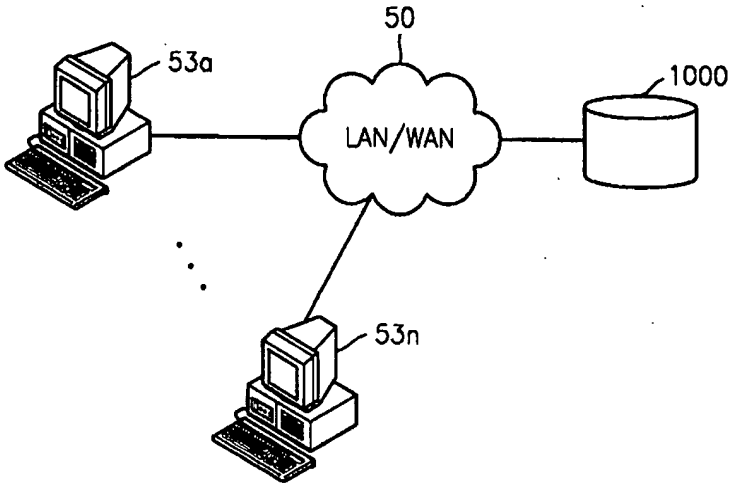
도면 3


30
10
31
11
15
어플리케이션
정보 요청지
어플리케이션
정보 제공 서버
어플리케이션
정보 DB
32
12
16
어플리케이션
페이지 로더
어플리케이션
페이지 제공 서버
어플리케이션
페이지 DB
17
33
13
핸들러
실행파일 DB
컴포넌트
로더
컴포넌트
제공 서버
18
컴포넌트
실행파일 DB

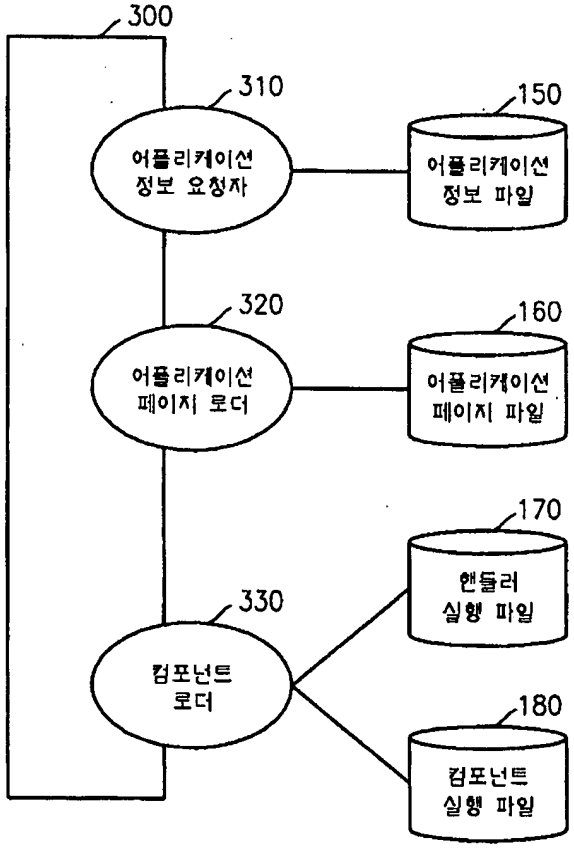
도면 4


100
101
102
103

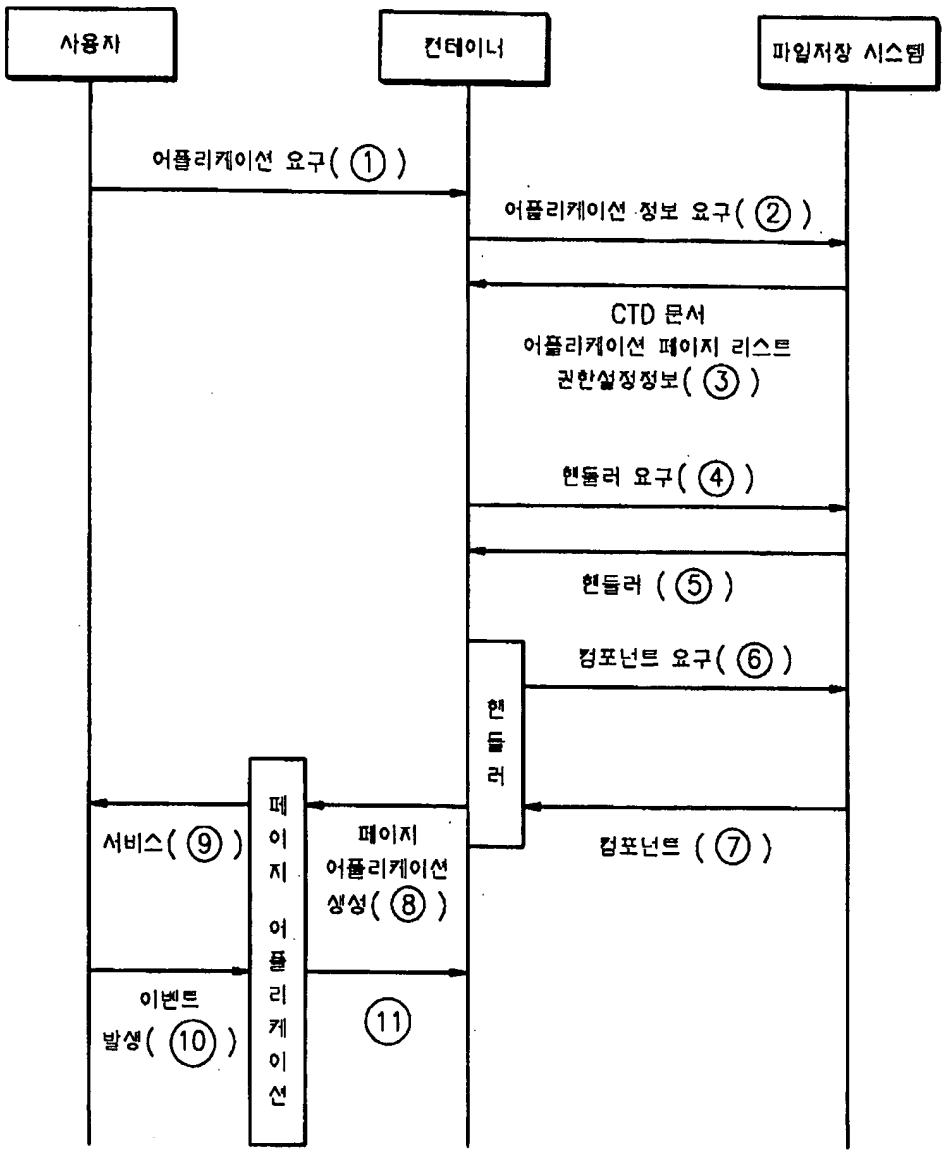
도면 5


50
53a
LAN/WAN
1000
53n

도면 6


300
310
어플리케이션
정보 요청자
150
어플리케이션
정보 파일
320
어플리케이션
페이지 로더
160
어플리케이션
페이지 파일
170
핸들러
실행 파일
330
컴포넌트
로더
180
컴포넌트
실행 파일

도면 7


사용자
컨테이너
파일저장 시스템
어플리케이션 요구(①)
어플리케이션 정보 요구(②)
CTD 문서
어플리케이션 페이지 리스트
권한설정정보(③)
핸들러 요구(④)
핸들러 (⑤)
컴포넌트 요구(⑥)
핸들러
서비스(⑨)
페이지 어플리케이션
생성(⑧)
컴포넌트 (⑦)
페이지 어플리케이션
이벤트
발생(⑩)
⑪

Laid-Open Publication Number: KR2001-0113472
Date: December 28, 2001

ABSTRACT

The present invention relates a method and apparatus defining a component model for creating dynamic documents in a Markup language format in a distributed data processing system. Static contents in the markup language format are located on the document by using a function call mechanism. The function call mechanism includes a function name and an actual parameter value that corresponds to a formal parameter that defines a function. An actual parameter value is provided through an electronic document access mechanism while accessing the dynamic electronic documents including the function call, and the function call is replaced with the markup language that contains a return value of the function call. The document then is formatted and retuned to be displayed. The document is formatted by applying a style sheet to the document.

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.		(11) 공개번호	특2001-0113472
G06F 17/00		(43) 공개일자	2001년12월28일
(21) 출원번호	10-2001-0028873		
(22) 출원일자	2001년05월25일		
(30) 우선권주장	09/596,897 2000년06월19일 미국(US)		
---	---
(71) 출원인	인터내셔널 비지네스 머신즈 코포레이션, 포만 제프리 엘 미국 000-000 미국 10504 뉴욕주 아몬크
(72) 발명자	암스덴제임스레이몬드 미국 미국노스케롤라이나주27513캐리클리어스카이코트110
(74) 대리인	허정훈 김성택
---	---
(77) 심사청구	없음

(54) 출원명 분산 데이터 처리 시스템에 있어서 동적 문서 생성용콤포넌트 모델 정의 장치 및 방법

요약

본 발명은 마크업 언어로된 동적 전자 문서를 생성하는 데이터 처리 시스템 내의 방법 및 장치이다. 마크업 언어의 정적 콘텐트가 전자 문서에 위치한다. 함수 호출 메커니즘을 마크업 언어을 이용하여 전자 문서에 위치시키는데, 이 함수 호출 메커니즘은 함수 호출에 이용되는 함수명을 포함하고, 함수 정의의 형식 파라미터에 대응되는, 공급되는 실제 파라미터 값도 포함한다. 함수 호출을 포함하는 동적 전자 문서를 접근하고, 문서 접근 메커니즘을 통해 그들의 실제 파라미터에 대한 값을 제공하는 것에 반응하여, 내장된 함수가 호출되고, 함수 호출이 반환 값을 보유한 마크업 언어로 대체된다. 그러면, 문서가 포맷되어 표시를 위해 반환될 것이다. 이 포맷은 문서에 스타일쉬트를 적용함으로써 이루어질 수 있다

.

대표도

도1

명세서

도면의 간단한 설명

도면의 간단한 설명

본 발명의 신규한 특징은 특허 청구 범위에 기술된다. 그러나, 본 발명 자체, 그리고 바람직한 실시예와 추가의 목적 및 효과는 첨부 도면과 상세한 설명에 의하여 더욱 자세히 이해될 수 있다.

도 1은 본 발명이 구현되는 분산 데이터 처리 시스템을 나타내는 도식도.

도 2는 본 발명의 양호한 실시예에 따라 서버로서 구현될 수 있는 데이터 처리 시스템의 블록도.

도 3은 본 발명의 양호한 실시예에 따라 동적 확장 마크업 언어(DXML) 문서에 이용되는 콤포넌트를 나타내는 도면.

도 4는 본 발명의 양호한 실시예에 따른 함수 정의 도면.

도 5는 본 발명의 양호한 실시예에 따른 함수 정의를 수용하는 문서를 나타내는 도면.

도 6은 본 발명의 양호한 실시예에 따른 함수 호출용 코드를 나타내는 도면.

도 7은 본 발명의 양호한 실시예에 따라, 실행 요소를 통해, 함수 정의를 포함하지 않는 구현으로 원함수(native function) 실행을 위한 코드의 예를 나타낸 도면.

도 8은 본 발명의 양호한 실시예에 따라 속성 값 반환용 구문(syntax)의 예를 나타낸 도면.

도 9는 본 발명의 양호한 실시예에 따라 도 3에 도시된 함수의 실행에 이용되는 절차를 나타내는 플로우차트.

도 10은 본 발명의 양호한 실시예에 따라 다양한 콤포넌트 사이의 데이터 흐름을 나타내는 도면.

도 11은 본 발명의 양호한 실시예에 따라 HTML 폼을 표출하는 데 이용될 수 있는 XSL 스타일쉬트를 이용할 때 함수 정의를 임포트하는 XML 문서의 예를 나타낸 도면.

도 12A 및 12B는 본 발명의 양호한 실시예에 따라 도 11에 도시된 XML 문서를 렌더링하는 데 이용되는 XSL 스타일쉬트의 예를 도시한 도면.

도 13A 및 13B는 본 발명의 양호한 실시예에 따른 실행 요소를 포함하는 XML 문서를 도시한 도면.

도 14A 및 14B는 본 발명의 양호한 실시예에 따라 도 13A 및 13B의 XML 문서를 표출하기 위한 XSL 스타일쉬트를 나타내는 도면.

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

1. 기술 분야

본 발명은 개선된 데이터 처리 시스템에 관한 것으로서, 특히 동적 문서(dynamic document) 생성용 방법 및 장치에 관한 것이다. 더 상세하게는 본 발명은 분산 데이터 처리 시스템에서 동적 XML 문서용 재사용 기능 제공에 관한 것이다.

2. 종래 기술

"인터네트워크"라고도 칭해지는 인터넷은, 데이터 전송 및 송신 네트워크로부터의 프로토콜에 의해 수신 네트워크에서 이용되는 메지시(가능하면 패킷)의 변환을 다루는 게이트웨이에 의하여 함께 결합되는, 상이할 수 있는 컴퓨터 네트워크들의 집합이다. 인터넷(Internet)의 'I'가 대문자로 쓰이면, TCP/IP 프로토콜 수트를 수용하는 네트워크 및 게이트웨이의 집합을 의미한다.

인터넷은 정보 및 오락의 근원으로서 문화적으로 확고하게 되고 있다. 많은 기업이 그들의 마케팅 노력의 필수 요소로서 인터넷 사이트를 생성하여, 자신들이 제안한 상품 및 서비스의 소비자에게 알리고, 브랜드 가치를 생성시키는 기타 정보를 제공한다. 연방 정부, 주 정부 및 지방 정부의 많은 기관들도 정보용으로 인터넷 사이트를 채택하고 있다. 더욱이, 인터넷은 상업 거래용의 매체로서 점점 보편화되어 가고 있다.

현재, 대부분 채택되는 인터넷상의 데이터 전송 방식은 간단히 '웹'이라고도 칭해지는 월드 와이드 웹 환경을 채택한다. 웹 환경에서, 서버 및 클라이언트는 하이퍼텍스트 트랜스퍼 프로토콜(HTTP) - 다양한 데이터 객체 또는 파일(문자, 정영상, 음성, 동영상 등)의 전송 취급용으로 알려진 프로토콜 - 을 이용하여 데이터 트랜잭션을 수행한다. 다양한 데이터 파일내의 정보는 종종 표준 페이지 기술 언어, 즉 하이퍼텍스트 마크업 랭귀지(HTML)에 의하여 사용자에게 표현되도록 포맷된다. 기본 표현 포맷에 부가하여, HTML은 개발자에게 유니폼 리소스 로케이터(URL)에 의해 식별되는 다른 웹 자원으로의 '링크'를 특정하는 것을 허용한다. URL은 특정 정보로의 통신 경로를 정의하는 특별한 구문 식별자이다. 클라이언트에게 접속 가능한 각 논리 블록 - 페이지 또는 웹 페이지라 칭함 - 은 URL에 의하여 식별된다. URL은 웹 브라우저를 이용한 정보 탐색 및 접근에 보편적이고 일관된 방법을 제공한다. 브라우저는 클라이언트 머신에서 URL에 의하여 식별된 정보에 대한 요청을 제출할 수 있는 프로그램이다. 웹 상에서의 정보의 검색은 일반적으로 HTML-호환성 브라우저로서 이루어진다. 또한, 인터넷은 브라우저 사용자로의 어플리케이션의 전송에도 널리 이용된다.

어플리케이션은 애플릿(applets) - 웹상에서 HTML 문서내의 객체로서 내장될 수 있음 - 이라고 알려진 스크립트 및 프로그램을 이용하여 웹상에 제공된다. 애플릿은 자바 프로그램인데, 자바 프로그램은 자신들이 나타나는 HTML 페이지와 함께 자바를 지원하는 브라우저로 투명하게 다운로드될 수 있는 것이다. 이들 자바 프로그램은 네트워크 및 플랫폼 독립적이다.

웹 페이지와 같은 문서와 웹 어플리케이션을 이용하여 기업 및 기타의 조직은 웹에서 이용되는 어플리케이션을 제공해왔다. 오늘날 웹 어플리케이션의 업무 논리는 HTML에 내장된 스크립트 언어와 공통 게이트웨이 인터페이스(CGI), 인터넷 서버 어플리케이션 프로그래밍 인터페이스(ISAPI), 자바 서블릿(servlets), 웹서버 플러그-인, 분산 콤포넌트 객체 모델(DCOM), 엔터프라이즈 자바빈(EJB) 또는 공통 객체 요청 브로커 구조(Common Object Request Broker Architecture; CORBA) 객체를 통한 웹서버의 확장을 조합하여 수행된다. 이는 개발자가 어플리케이션의 각 모듈에 통합해야만 할 대단히 많은 기술때문에, 웹 어플리케이션 개발자에게 복잡한 해결책을 제시한다. HTML 및 동적 HTML(DHTML)은 임시변통적이며, 어떠한 정형화된 컴퓨팅 모델이 아니다. 그 결과, 업무 논리, 뷰 렌더링(view rendering) 및 동적 사용자 인터페이스를 제공하기 위한 스트립트를 이용하는 HTML 문서는 개발하기 어렵고, 브라우저에 민감하며, 유지하기 어렵고, 재사용 기회의 제한이 있다.

확장가능 마크업 언어(Extensible Markup Language; XML)는 웹 상의 문서 교환을 위한 또 다른 마크업 언어이다. 현재 동적 XML 문서의 생성 또는 XML에서의 스크립트 지원에 대한 표준은 없는 상태이다. 확장가능 스타일쉬트 언어(extensible stylesheet language; XSL)는 하나의 XML 문서를 다른 XML 문서로 변환하는 데 이용될 수 있지만, 이것은 1회성(one-shot) 처리이다. 변환의 수행 결과는 정적 문서이다. 또한, XSL는 유효한 스크립팅 솔루션을 제공하지 않는다.

문서에 기능성을 부여하기 위하여 제안된 몇몇의 메카니즘은 DHTML, 액티브 서버 페이지(Active Server Page; ASP), 자바 서버 페이지(JSP), 및 마이크로소프트 인터넷 익스프로러 5(IE5)의 작용(behavior)이다. DHTML은 예측이 불가능한 구조가 될 수 있고 유지하기 어려운 애드-혹 스크립팅(ad-hoc scripting)을 지원한다.

ASP 및 JSP는 동일한 문제를 해결하지만, 템플릿 언어를 이용하여 이를 행한다. ASP 및 JSP는 함수 호출을 위해서는 문서 저작이 프로그래밍 영역으로 전환될 것을 요한다. HTML 저작 도구(authoring tool)는 APS 및 JSP의 개발을 단순화하는 데 도움을 주지만, 이러한 툴은 여전히 많은 상이한 영역이 혼재되어 있다.

마이크로소프트 코포레이션의 IE5 작용은 캐스캐이딩 스타일쉬트(Cascading style sheet; SCC)로써 스크립트 함수를 XML 요소와 연관시킨다. XML 요소에 부가된 작용은 그들의 스키마와 일치하지 않는다. 동일한 XML 요소가 어떤 다른 CSS 스타일쉬트로부터 계승된 완전히 상이하고 부적절한 작용을 가질 수 있다. IE5 내의 작용은 원격 프로시저 콜을 지원하지 않는다. IE5의 작용은 단지 스타일쉬트 저작자가 메쏘드, 계산된 프로퍼티, 및/또는 이벤트를 임의의 HTML 도는 XML 요소와 연관시키게 하고, 별개의 문서에서 그것들을 정의하도록 한다.

발명이 이루고자 하는 기술적 과제

따라서, 분산 데이터 처리 시스템에서 문서를 생성하고 공급하는 방법 및 장치를 개선시키는 것은 매우 이로운 것이며, 본 발명은 이러한 방법 및 장치를 제공하려는 것을 그 목적으로 한다.

발명의 구성 및 작용

발명의 요약

본 발명은 마크업 언어로 동적 전자 문서를 생성하는 데이터 처리 시스템 내의 방법 및 장치를 제공한다. 마크업 언어 내의 정적 콘테트가 전자 문서에 위치한다. 함수 호출 메카니즘이 마크업 언어를 이용하여 전자 문서에 위치하는데, 함수 호출 메카니즘은 함수 호출을 위한 함수명 및 파라미터를 포함한다.

함수 호출 메카니즘 활성화에 응답하여, 함수 실행을 요청하는 마크업 언어(즉, XML)가 그 결과를 표시하는 마크업 언어로 대체된다. 예를 들어, HTML과 같은 마크업 언어 문서는 생성된 후, 함수 정의를 살펴보고 그 함수에 대한 파라미터의 입력을 지시하기 위한 형식을 표출하는 요청에 따라 표출될 수 있다. 이러한 파라미터가 수신되면, 그 함수를 호출하는 다른 XML 문서에 요청을 포스팅함으로써 함수가 호출된다. 위 예에서 함수는 함수 호출을 요청하는 마크업 언어를 포함하는 문서를 검색하고, 파라미터를 그 함수에 전달하고, 그 함수를 실행함으로써 호출된다. 함수의 실행에 따른 결과는 함수 호출 요청을 반환 값을 나타내는 마크업 언어 결과로 교체한다. 그러면, 문서는 포맷되고 표현을 위하여 반환된다. 이 포맷팅은 문서에 스타일쉬트를 적용함으로써 달성된다.

실시예의 상세한 설명

1. 환경

도 1은 본 발명이 구현될 수 있는 분산 데이터 처리 시스템의 도식화한 도면이다. 분산 데이터 처리 시스템(100)은 본 발명이 구현될 수 있는 컴퓨터 네트워크이다. 분산 데이터 처리 시스템(100)은 네트워크(102)를 포함하며, 이 네트워크는 분산 데이터 처리 시스템(100)내에 상호 접속된 다양한 장치 및 콤포넌트를 사이의 통신 링크를 제공하는 데 이용되는 매체이다. 네트워크(102)는 와이어 또는 광섬유 케이블과 같은 영구적 연결 또는 전화 접속을 통한 일시적 연결을 포함할 수 있다.

본 실시예에서, 서버(104)는 저장 유닛(106)과 함께 네트워크(102)에 접속된다. 부가하여, 클라이언트(108, 110, 112)도 네트워크(102)에 접속된다. 이들 클라이언트(108, 110, 112)는, 예컨대 개인용 컴퓨터이거나 네트워크 컴퓨터이다. 이러한 응용의 목적의 관점에서는, 네트워크 컴퓨터는 네트워크에 결합되어 네트워크에 결합된 다른 컴퓨터로부터의 프로그램 또는 다른 응용 어플리케이션을 수신하는 임의의 컴퓨터이다. 이 실시예에서, 서버(104)는 부트 파일, 운영체제 이미지, 및 클라이언트(108-112)로의 어플리케이션과 같은 데이터를 제공한다. 클라이언트(108, 110, 112)는 서버(104)에게 요청한다. 분산 데이터 처리 시스템(100)은 도시되지 않은 추가의 서버, 클라이언트 및 다른 장치를 포함할 수 있다. 설명되는 실시예에서, 분산 데이터 처리 시스템(100)은 상호 통신용 TCP/IP 프로토콜 수트를 이용하는 네트워크 및 게이트웨이의 범세계적 집합체를 나타내는 네트워크(102)를 구비하는 인터넷이다. 인터넷의 핵심은 데이터 및 메시지를 라우트하는 수천 개의 상업, 정부, 교육 및 기타 용도의 컴퓨터 시스템으로 이루어지는 주요 노드 또는 호스트 컴퓨터사이의 고속 데이터 통신선인 백본이다. 물론, 분산 데이터 처리 시스템(100)은 예컨대 인트라넷, LAN, 또는 WAN과 같은 많은 수의 상이한 네트워크로 구현될 수도 있다. 도 1은 예시일 뿐이며 본 발명의 구조를 제한하려는 것은 아니다.

본 발명은 분산 데이터 처리 시스템(100)에 구현되어 분산 문서에 콤포넌트 모델을 제공하고 재사용가능 콤포넌트를 이용하여 문서 내에 동적 로직 또는 함수를 제공할 수 있다. 이러한 실시예에서, 본 발명에 의해 제공되는 모델은 문서에 집중되고 XML에 기초한다. 추가적으로, 이 모델은 언어 중립적(language neutral), 위치 독립적, 플랫폼 독립적이다. 문서는 클라이언트(108-122)와 같은 클라이언트상의 다양한 사용자에게 분배된다. 함수는 이들 클라이언트 또는 서버(104)와 같은 다른 시스템에 위치할 수 있다. 네트워크(102)는 추가의 서버를 포함할 수도 있다. DXML 문서는 상이한 어플리케이션을 위해 서버간에 전송될 수 있다.

도 2를 참조하면, 도 1의 서버(104)와 같은 서버로서 구현될 수 있는 데이터 처리 시스템의 블록도가 본 발명의 양호한 실시예에 따라 묘사되어 있다. 데이터 처리 시스템(200)은 시스템 버스(206)에 연결된 복수 개의 프로세서(202, 204)를 포함하는 대칭형 멀티프로세서(SMP) 시스템일 수 있다. 대안으로서, 단일 프로세서 시스템이 채택될 수도 있다. 또한 로컬 메모리(209)에 인터페이스를 제공하는 메모리 콘트롤러/캐쉬(208)가 시스템 버스(206)에 연결될 수 있다. 메모리 콘트롤러/캐쉬(208) 및 I/O 버스 브릿지(210)는 도시된 대로 통합될 수 있다.

I/O 버스(212)에 연결된 PCI 버스 브릿지(214)는 PCI 로컬 버스(216)로의 인터페이스를 제공한다. 복수 개의 모뎀이 PCI 버스(216)에 연결될 수 있다. 통상적인 PCI 버스 구현은 4 개의 PCI 확장 슬롯 또는 애드-인(add-in) 콘넥터를 지원한다. 도 1의 네트워크 컴퓨터(108-112)로의 통신 링크는 모뎀(218)과, 애드-인 보드를 통하여 PCI 로컬 버스(216)에 연결된 네트워크 어댑터(220)를 통하여 제공될 수 있다.

부가적인 PCI 버스 브릿지(222, 224)는 추가적인 모뎀 또는 네트워크 어댑터를 지원할 수 있는 부가적인 PCI 버스(226, 228)용 인터페이스를 제공한다. 이러한 방식으로, 데이터 처리 시스템(200)은 복수 개의 네트워크 컴퓨터에 대한 연결을 허용한다. 메모리-맵 그래픽 어댑터(memory-mapped graphic adapter)(230) 및 하드디스크(232)도, 도시된 것처럼 직접적 또는 간접적으로 I/O 버스(212)에 연결될 수 있다.

이 분야의 통상의 지식을 가지는 자라면 도 2에 도시된 하드웨어는 매우 다양할 수 있다는 것을 알 것이다. 예컨대, 광학 디스크 드라이브와 같은 기타 주변 장치도 추가되거나 도시된 하드웨어 대신에 이용될 수 있다. 묘사되는 실시예는 본 발명에 대한 구조적 제한을 암시하는 것은 아니다.

도 2에 도시된 데이터 처리 시스템은, 예컨대 미국 뉴욕 아몬크 소재의 인터내셔널 비지네스 머신즈 코포레이션의 상품인, 어드밴스트 인터액티브 이규제큐티브(AIX) 운영 체제를 탑재한 IBM RISC/System 6000 시스템일 수 있다. 더욱이, 도 2에 도시된 데이터 처리 시스템은 클라이언트에 구현될 수도 있다. 클라이언트는, 예컨대 노트북 컴퓨터, PDA, 파지용 컴퓨터, 키오스크 또는 웹 어플라이언스와 같은 다양한 형태일 수 있다.

2. DXML

본 발명은 웹상의 이-비지니스와 같은 어플리케이션을 지원하기 위하여 재사용가능 콤포넌트의 상태와 작용(state and behavior)을 캡슐화한 콤포넌트 모델이 필요하다는 것을 인식한다. 콤포넌트 모델이 없다면, 클라이언트 및 비즈니스 어플리케이션은 복잡해지고 작성하는 데 비용이 드는데, 이는 각 어플리케이션이 데이터를 분석하고 이것의 의미 및 작용을 재생성해야 하기 때문이다. 상호작용성은 통합하기 어려운 자동 사일로(silos of automation)를 생성하는 데이터의 의미를 재창조하는 어플리케이션에 의해 손상된다. 데이터 무결성도 각 어플리케이션이 원시 데이터를 접근하여 이를 상이하게 번역할 수 있으므로 손상될 수 있다. 최종적으로, 재사용은 현격하게 제한된다.

본 발명의 콤포넌트 모델은 개발자들에게 익숙하고 공통 업무와 일치되는 방식으로 문서 생성 가능성 및 기능성을 부가한다. 이 메카니즘은 콤포넌트 모델이 폭넓은 저작 사회(autoring community)에 접근 가능하게 한다. 본 발명의 메커니즘은 업무에 특정된 기술의 사용을 극대화하기 위하여, 별개의 문서 저작, 문서 스키마 저작 및 문서 레이아웃 저작의 규칙을 지킨다. 본 명세서에서 사용되는 '스키마'는 문서 구조의 정의다.

본 실시예에서는, 본 발명은 XML 기반의 문서 중심 콤포넌트 모델에 집중한다. 동적 XML 문서의 생성 또는 XML의 스크립팅 지원에 대해 현재로서는 어떤 표준이 없다. XLS는 XML 문서를 또 다른 XML 문서로 변환하는 데 이용될 수 있으나, 이것은 1회성 처리(one-shot process) 이다. 일단 변환이 완료되면, 그 결과는 정적 문서이다. XSL은 효율적인 스크립팅 솔루션을 제공하지도 못한다.

본 발명은 웹용 컴포넌트 모델을 위한 방법, 장치 및, 컴퓨터에 구현된 명령어를 제공한다. 설명되는 실시예에서, 웹 콤포넌트 모델은 XML 체계 및 함수 정의로부터 형성된다. 본 실시예에서, 콤포넌트 모델은 동적 XML(DXML)이다. 이들 실시예에서, DXML은 XML과 스크립팅의 조합이다. XML 요소는 태그명, 속성 집합, 및 콘텐트 모델을 통한 제한을 포함하는 다른 요소와의 연관, 및 XLink 및 XPointer를 통한 일반 연관을 구비한다. DXML은 요소가 다른 XML 요소 또는 문자 데이터를 반환하는 메쏘드를 가질 수 있게 함으로써 XML을 확장한다. DXML은 XML 요소상의 메쏘드 정의용 XML 스키마로의 확장, 요소 메쏘드의 구현 로직 지정용 공통 스크립팅 언어, 함수 호출을 위한 심플 오브젝트 억세스 프로토콜(SOAP), 및 웹용 콤포넌트 모델에 비중을 두는 문서 정의용 XML을 조합한다. 본 실시예에서, 새로운 XML 메커니즘이 함수 호출을 위하여 이용된다. 물론, SOAP 및 다른 공지의 함수 호출 메커니즘이 본 발명의 양호한 실시예에 따른 함수 호출을 위하여 이용될 수 있다.

본 발명의 한 가지 이점은 프로그래밍 언어를 사용할 필요 없이 자바 및 다른 콤포넌트 모델처럼 함수를 스크립트하여 정의된 작용을 이용하여 XML 문서 저작을 가능하게 하는 데 집중한다는 것이다.

공식적인 방식의 표준 스크립팅 언어를 XML과 조합함으로써, DXML은 전술한 웹 콤포넌트 모델의 요구를 충족시킨다. 본 발명의 웹 콤포넌트 모델은 XML에 기초하고 공통 스크립팅 언어를 이용하여 함수를 구현하는데, 이 모델은 태그 언어 및 스크립트를 이용하는 현재의 HTML 저자들에게 익숙할 것이다. 스키마와 로직 개발, 업무 데이터를 분리하고 이들을 공식화된 프로그래밍 모델을 통하여 통합할 때 명령어를 공급함으로써, 웹 어플리케이션의 복잡성이 감소되고, 유지보수성이 증가한다. DXML은 함수들이 스크립트 언어나 어떤 배포된 서비스의 자연적 방법으로 구현되는 것을 허용한다. DXML은 XML을 이용하여 스크립트나 분산된 콤포넌트의 서비스를 서비스의 구현에 이용된 콤포넌트 모델에 독립적으로 호출한다. 필수적으로, DXML은 XML 요소의 메쏘드 개념을 공통 스크립트 언어 및 XML을 이용하여 호출된 원격 프로시저 콜과 조합함으로써 웹 어플리케이션에 이상적인 콤포넌트 모델을 생성한다.

3. 데이터 플로우

도 3은 본 발명에 따르는 동적 확장 가능 마크업 언어(DXML) 문서 처리에 이용되는 데이터 흐름을 도시한 도면이다. 본 실시예에서, 클라이언트(300)는 DXML 문서를 이용하여 서버(302)로부터 정보를 접근하고 요청한다. 이 정보는, 예컨대 과급여 고용인을 식별하는 요청일 수 있다. 클라이언트(300)와 서버(302) 사이의 통신은 HTTP를 이용하여 이루어진다.

클라이언트(300)상의 브라우저는 함수 실행의 요청을 생성하는 데 이용된다. 특히, 클라이언트(300)는 서버(302) 내의 함수(306)를 겟(get)하도록 전송되는 DXML 문서 URL을 생성한다.(단계 m1). 예를 들어, 클라이언트(300)상의 사용자는 클라이언트(300) 내의 웹 브라우저(304)에 표출되는 웹 페이지 내의 링크 또는 하이퍼링크를 선택할 수 있다. 요청 링크는 회사의 과급여(過給與) 고용인을 식별하는 함수와 같은 함수를 호출하는 것일 수 있다. 그러한 링크의 선택에 반응하여, 이 특정 함수에 연관된 URL이 '겟' 함수(306)로 반환된다.

응답에 있어서, 겟 함수(306)는 서버(302) 내의 웹 서버 저장소(308)로부터의 URL에 대응하는 DXML 함수를 구현하는 데 이용되는 XML 문서를 획득한다(단계 m2). 이 경우, 서버(302)는 모든 XML 문서(MIME 타입 텍스트/xml 또는 어플리케이션/xml이 있는 문서)를 XML/XSL이 가능하지 않은 브라우저에 표출되도록 번역 함수(310)를 이용하여 HTML로 번역한다. 모든 문서는 저장소로부터 얻을 수 있다. 사용되는 웹 서버에 따라 특정 구성이 이용된다. 예를 들어, IBM WebSphere AppServer 및 서블릿 필터링이 번역을 행하는 데 이용될 수 있다.

XSL을 이용하여 XML을 번역할 수 있는 브라우저(예컨대, IE5)가 채택되면, 서버 에 이 번역 단계는 필요하지 않다. 이 경우, 겟 함수(306)의 결과는 번역 함수(310)과 연쇄된다(단계 m3). 번역 함수(310)는 DXML 문서를 참조되는 XSL 스타일쉬트(DXML에서 참조됨) 내의 명령어에 기초하여 HTML 문서로 변환한다. 그 다음, 이 HTML 문서는 겟 함수(306)의 결과로서 웹 브라우저(304)로 반환된다(단계 m4).

DXML 함수 문서의 획득에 응답하여, 이 DXML 문서는 DXML 문서를 HTML 문서로 번역하거나 변환하는 번역 함수(310)에 전달된다. 이것은 웹 서버 저장소(308)로부터 XSL 스타일쉬트를 얻음으로써 이루어진다(단계 m3). 스타일쉬트는 DXML 문서에 적용되어 HTML 문서를 생성하게 한다. 이 DXML 문서는 번역 함수(310)에 의해, 원래 요청에 응답하여 HTML 문서를 클라이언트로 반환하는 겟 함수(305)로 반환된다(단계 m6).

이 HTML 폼은 사용자에게 값을 입력하라고 지시하는 용도의 인터페이스를 HTML 폼을 통하여 제공하고, DXML 문서가 다른 용도를 위하여 어떻게 번역되고 검토되는 지를 나타내는 예시를 제공한다. 클라이언트(300)에서, 이 HTML 폼은 웹 브라우저(304)에 표출되고, DXML 함수에 의하여 요구되는 파라미터를 요청하는 데 쓰인다. 웹 브라우저(304)에서, HTML 문서가 작업 유형 및 최대 급여 등의 값을 위한 필드 및/또는 프롬프트 를 포함하며 함께 표출된다. 이들 필드 및 프롬프트는 함수 정의를 고찰함으로써 유도될 수 있고, XSL 스타일쉬트에 의하여 생성될 수 있다. 이 HTML 문서에 대한 사용자 입력에 반응하여, DXML 문서 요청 및 질의열이 포스트 함수(312)로 전달된다(단계 m7). 포스트 함수(312)는 웹 서버 저장소 내의 문서를 접근하는 데 쓰인다. 겟 함수(306) 및 포스트 함수(312)는 본 발명의 양호한 실시예에 사용되는 데이터 플로우를 명확하게 설명하려는 목적으로 별개의 함수로서 설명되고 있다. 이들 콤포넌트는 구현에 따라서 또는 특정 요청에 따라 실제로는 동일한 콤포넌트일 수 있다. 이들 요청은 실행 함수(314)로 전달된다(단계 m8).

실행 함수(314)는 포스트 함수(312)로부터 수신된 요청 및 질의 파라미터에 기초하여 DXML 문서를 실행하는 데 쓰인다. 포스트 함수는 클라이언트로부터의 HTTP 웹 서버 내의 프로세스로의 접근을 허용하는 메커니즘을 제공하는 데 이용된다. 이들 예에서, 겟 함수 또는 포스트 함수는 HTTP를 이용하여 클라이언트와 통신한다. 클라이언트로부터 수신된 HTTP 요청은 서버(302)내에서 처리되기 위하여 적절한 폼으로 변환된다. 이들 요청에 대한 응답은 겟 함수 또는 포스트 함수에 의해 수신되어 클라이언트(300)로 재전달되기 위하여 HTTP 형식으로 재변환된다. 요청 및 질의 파라미터의 수신에 반응하여, 실행 함수(314)는 웹 서버 저장소(308)로부터 적절한 DXML 문서를 검색한다(단계 m9).

추가적으로, (만약 있다면) XML 스키마 및 DXML 함수 정의 또한 웹 서버 저장소(308)로부터 검색된다(단계 m10). 실행 함수(314)는 DXML 문서를 판독할 것이다. DXML 문서의 판단에 반응하여, 호출 함수로의 요청이 문서 마크업 언어로 식별된다. 이 다음, 함수가 실행되고 그 결과가 함수의 결과를 나타내는 마크업 언어로써 DXML 문서에 위치하게 된다. 이 교체는 DXML 문서를 XML 문서로 바꾼다.

다음, XML 문서의 최종 폼이 번역 함수(316)로 전달된다(단계 m11). 번역 함

수(316)는 적절한 XSL 스타일쉬트를 웹 서버 저장소(308)로부터 얻는다(단계 m12). 이 스타일쉬트는 XML 문서에 적용되어 HTML 문서를 생성하게 한다. HTML 문서는 포스트 함수(312)로 전달된다(m13). 이 HTML 문서는 클라이언트(300)로 전달된다(m14). 클라이언트(300)에서는, 결과를 포함하는 HTML 문서가 웹 브라우저(304)에 표출된다. 첨언하자면, 브라우저가 XSL 스타일쉬트로써 XML을 처리할 수 있다면, XML에서 HTML로의 번역 단계는 필요하지 않다.

본 발명의 DXML 모델은 문서 함수(즉, 자유 함수) 및 요소 멤버 함수 모두를 정의한다. 문서 함수는 어떤 요소에도 속하지 않는 함수이다. 이들 함수는 독립적이고 그들이 호출된 문서와 그들의 실제 파라미터 내에서 정의된다. 이들 실시예에서, 문서 함수는 그들이 사용되는 XML 문서-XML 엔티티 선언을 이용하는 XML 문서에 포함됨- 또는 XML 스키마에서 정의될 수 있으나, 스키만 내의 어떠한 특정 요소에 연관되지는 않는다. 따라서, 문서 함수는 XML 스키마를 필요로 하지 않고, 단지 XML 문서의 형식이 잘 갖춰지기만 하면 된다.

실시예에서, 문서 함수는 C++의 자유 함수 또는 C, BASIC, 자바스크립트 및 Ada와같은 절차적 언어의 함수에 대응한다. 함수가 실행될 수 있는 공유 콘텍스트를 대표하는 클래스가 문서에 대응될 때, 문서 함수는 C++, 자바 및 스몰톡(Smalltalk)의 클래스 함수와 유사하다.

요소 멤버 함수는 XML 스키마로의 확장부를 이용하여 정의된다. XML 내의 요소는 이름, 약간의 속성, 자신의 콘텐트 모델 내의 요소와의 제한 연관 및 XLink 및 XPointer를 통한 일반 연관을 포함하는 다른 요소와의 연관을 포함한다. DXML은 XML 스키마로 확장하여 스키마 저작이 요소 정의 내에 멤버 함수를 포함하는 것을 허용함으로써 작용을 포함한다. DXML은 XML로 확장하여 요소가 UML, C++ 또는 자바 클래스, 또는 COM 및 CORBA 객체에 대응되도록 한다. DXML 함수는 함수로부터 그 상태 천이(생명 주기)에 영향을 받는 요소와 함께 작용을 캡슐화한다. 이로써, XML은 웹 상의 서비스를 기술하는데 이용될 수 있고, XML 문서를 이용하여 문서 저작에 의하여 이들 서비스의 호출을 함께 스크립트 하도록 하여 업무상 문제를 해결하고 다른 프로그램과 결과를 통신하고 웹 브라우저에 이를 표출할 수 있게 한다.

이들 실시예에서, 함수는 가능한 한 XML로 기술되어 검토에 용이하게 하고, XML 문서 저작자에게 쉽게 이용할 수 있게 하고, 언어 중립적이 되게 한다. 함수의 본체만이 언어 특정적이다.

4. 함수 정의

도 4는 DXML의 전반적 체계도 및 본 발명의 양호한 실시예에 따라 도시된 함수 정의도이다. DXML은 표준 XML 및 공통 스크립트 언어와 웹용 콤포넌트 모델을 만들기위한 함수 정의를 포함하기 위한 XML 스키마 확장부로 이루어진다. 함수 정의(400)는 함수명(402), 함수 설명(404), 형식 파라미터(406), 반환 값 타입(408), 및 함수 본체(410)으로 이루어진다. 이들 실시예에서, 형식 파라미터(406)는 파라미터명, 타입, 설명 및 기본 값을 가진다. 본 실시예에서는, 반환값(408) 타입은 XML 요소 또는 문자 데이터(CDATA)일 수 있다. 함수명은 XML 이름공간(namespace)에 의해 범위가 정해지고, 함수 파라미터는 그 이름이 연관된 함수에 의해 범위가 정해진다. 만일 그 함수가 스크립트 언어를 이용하여 구현되면, 함수 본체(410)는 스크립트 명령어를 포함할 수 있다. 대안으로서, 함수가 어느 곳이나 구현되면 함수 본체(410)는 빌 수도 있다. 이 경우, 함수는 그 이름으로 자신이 서버에 구현된 것을 충분히 식별할 수 있는 문서 함수일 수 있다. 다른 경우, HTML에 사용되는 것과 같은 실행 요소의 href 속성이 함수를 구현하는 원격 객체를 식별하는 데 이용될 수 있다.

다음의 도 5는 본 발명의 양호한 실시예에 따른 문서 함수를 도시한 도면이다. 이 실시예에서, 문서 함수(500)는 'overpaidEmployees'라는 이름이 붙여졌다. 이 문서 함수는 도 3의 DXML 함수 XML 문서에 대응한다. 문서(500)는 어떠한 XML 문서 또는 XML 스키마에 나타날 수 있는 문서 함수의 예를 포함한다. 본 실시예에서, 문서 함수(500)는 섹션(502)에 6개의 형식 파라미터를 구비하는데, 이중 4개는 데이터 베이스 접근에 필요한 것이다. 섹션(504)의 반환 값은 태그명 'overpaidEmployees'이 붙은 XML 요소이다. 섹션(506)의 함수 본체는 본체 요소의 언어 속성에 의해 나타나듯이 동적 구조적 질의 언어(SQL)를 사용하여 작성된 것이다.

만약 본체가 밖으로 벗어나야 할 많은 수의 XML 캐릭터를 포함한다면, 완전한 함수 본체를 CDATA 섹션에 포함시키는 것이 편리할 수 있다. 함수가 EJB, CORBA, 또는 DCOM 객체로서 구현되었다면, 언어 속성은 모(母; native) 값을 가질 것이고, 어느 곳에서건 구현이 될 때 그 본체는 빌 것이다. 형식 파라미터 타입 및 그들의 허용 가능 타입의 의미는 함수 본체 언어에 의해 정의되지만 XML 언어로써 기술된다. 본 실시예에서, 파라미터 타입은 SQL로부터 얻어진다. 함수 실행 엔진은 함수 본체 언어에 대해 타입이 명확한지를 결정한다.

이러한 방식으로, DXML은 예컨대 자바스크립트, VBScript, 자바, REXX, 및 Perl과 같은 많은 공통 스크립트 언어를 지원할 수 있다. 요소 멤버 함수는 XML 스키마에 유사하게 정의된다. 요소 멤버 함수와 문서 함수와의 차이는 요소와 함수와의 연과, 함수의 콘텍스트가 자체 수용 요소를 포함하는 것, 및 함수가 정의된 요소의 차일드로서 나타나기 위해서 함수의 호출이 필요한 것이다. DXML은 함수 정의에 XML을 이용하므로, DXML은 확장 가능하다. 구현은 속성 또는 DXML로의 콘텐트 모델 확장부와 같은 추가의 메타-데이터를 포함할 수 있다. 이러한 확장부를 알지 못하는 어플리케이션은 XML의 관례대로 이 추가의 요소를 단순히 무시한다. 함수 호출은 후속 섹션에 기술된다. 모든 함수는 함수를 구현할 때 이용된 언어가 아니라 XML을 이용하여 DXML 문서 내에서 호출된다. XML 문서 저작자는 함수의 구현, 작성된 언어, 또는 스크립트 언어인지의 여부 또는 분산 서비스로서 구현되었는 지에 대해서는 알 필요가 없다. DXML 문서가 접근되면, 문서 내의 함수 호출이 실행되고 그 결과로 대체된다. 모든 함수는 XML 요소 또는 CDATA를 반환한다. XML 요소를 반환하는 함수는 페어런트 요소의 콘텍스트 내에서 호출되어야 하는데, 페어런트 요소는 그 콘텐트 모델 내의 차일드로서 XML 요소를 반환하여 그 결과로서 명백한 XML 문서를 생성한다. DXML 문서는 문서 타입 정의(DTD) 또는 XML 스키마가 없으면, 문서는 단지 잘 형식화되면 되므로, 반환된 XML 요소는 임의의 XML 요소의 차일드로 여기질 수 있다. 함수가 CDATA를 반환하면, 데이터는 요소의 콘텐트 모델 내에 파스된 문자 데이터(parsed character data; PCDATA)를 허용하는 임의의 XML 요소에 나타날 수 있다. PCDATA는 XML 내 요소의 콘텐트일 수 있다. 대안으로서, 데이터는 속성 값을 제공하기 위하여 이용될 수 있다.

DXML 함수 호출용 실제 XML 구문은 메쏘드, 자체 파라미터, 및 반환값을 정렬하기 위해 XML을 이용하는, 클라이언트로부터 서버 어플리케이션으로의 원격 프로시저 콜 호출용 메커니즘과 일치되어야 한다. 실시예에 사용되는 호출은 분산 함수 호출용 신구문을 규정한다는 것을 의미한다기보다 이것이 동적 문서의 콘텍스트에서 어떻게 수행되는 지를 예시하는 것이다. 함수 호출 구문의 정확한 형식은, 이 실시예에서, 가능한 한 SOAP, WebBroker, 또는 웹 인터페이스 기술 언어(WIDL)를 정렬하는 현존의 XML 원격 프로시저 콜과 상응한다. 달리 말하면, 함수 호출은 XML이고 적어도 1) 실행될 함수명(필요한 이름공간 정보를 포함함); 2) 함수가 분산 서비스로서 구현되었다면 함수를 구현하는 객체에 대한 레퍼런스; 및 3) 실제 파라미터 값을 포함한다.

5. 함수 호출

도 6은 본 발명의 양호한 실시예에 따른 함수 호출용 코드를 나타낸 도면이다. 함수 호출(600)은 도 5의 함수(500)를 호출한다. 본 실시예는 부서의 업무 콘텍스트 내의 함수(500)을 호출한다. DXML은 요청 URI(Universal Resource Identifier)의 질의 파라미터가 DXML 메쏘드의 콘텍스트의 일부로서 사용되는 것을 허용한다. 본 실시예에서, 직무(job) 파라미터의 값은 요청 URI의 직무 파라미터에 의해 제공된다. 예를 들면 http://someDomain/companyStatistics?Job=MANAGER&salary=100000 이다. 급여 파라미터 값은 이 경우 상수이다. DXML 문서의 실행시, 섹션(602)의 실행 요소는 이 경우 $100,000 이상을 받는 관리자에 대한 정보를 수용하는 'overpaidEmployees" 요소인 함수의 결과로 대체된다.

필요하다면 그 값이 RMI, CORBA, 또는 DCOM 객체의 URI인 추가의 href 속성을 구비하며, 모 객체(native class)상의 메쏘드를 호출하는 것인 실행 요소를 이용함으로써 모 함수(native function)가 지원된다.

도 7은 본 발명에 따르는 실행 요소를 통한 모 함수의 실행용 코드를 예시한 도면이다. 본 실시예에서, 실행 요소(700)는 모 함수의 호출을 준비한다. 본 실시예에서, Iiop://someDomain/aCpmpany는 메쏘드 overpaidEmployees를 가지는 클래스의 인스턴스를 참조하는 URI이다. 이 유형의 호출 메커니즘은 스크립트 함수를 호출하고 함수가 구현된 곳과는 다른곳에 동일한 다른 객체상의 함수를 호출한다. 문서 저작자는 SQL, JavaScript, Java, 또는 웹 서버에서 가능한 DXML에 의해 지원되는 기타의 스크립트 언어를 이용하여 자신들만의 단순한 함수를 정의할 수 있다. 부가적으로, 문서 저작자는 또한 엔터프라이즈 자바빈 및 COM 객체의 함수와 같은 함수를 접근할 수 있다. 이 접근은 정확히 동일한 단순 메커니즘을 이용하여 달성될 수 있다. 본 발명의 함수 인터페이스는 여전히 DXML 함수로 정의된다. DXML 함수는 XML을 이용하여 호출된다. 차이점은 본체가 비었다는 점, 언어 = "native"인 점, 및 문서 저작자가 적절한 서비스 공급자의 이름을 제공한다는 점이다.

CDATA를 호출하는 DXML 함수는 또한 속성 값을 제공하기 위하여 이용될 수 있다. 도 8은 본 발명의 양호한 실시예에 따른 CDATA 반환용 구문의 예를 도시한 것이며, 이를 참조하여 설명한다.

DXML 함수가 실행되는 데는 몇 가지 방법이 있다. 예를 들어, 이들 DXML 함수는 웹 브라우저 플러그-인 또는 애플릿을 이용하여 클라이언트 상에서 실행되고, CGI 또는 서블릿 또는 임의의 기타 웹 서버 확장 메커니즘을 이용하여 서버 상에서 실행되거나, 어플리케이션 프로그램 내의 XML 문서 객체 모델(Document Object Model; DOM)을 통하여 호출된다. 서블릿은 인터넷 또는 인트라넷 HTTP 웹 서버에서 실행되는 자바로 작성된 소규모 프로그램이다. 실행의 시멘틱은 DXML 문서가 실행되는 위치에 무관하게 동일하다. DXML 문서 실행 위치의 차이는 성능에 있다. DXML 함수내의 처리가 단순한 스크립팅 또는 분산 서비스로의 접근이라면, DXML 문서는 하등의 성능 저하 없이 클라이언트에서 실행될 수 있다. 그러나, DXML 함수가 서버에 이미 존재하는 데이터를 접근하면, 성능 향상을 위해 데이터에 근접한 DXML 문서를 실행하는 것이 좋을 것이다. 실행된 요소의 "at" 속성은 함수가 어디에서 실행되어야 하는 지를 규정하기 위해 문서 저작자에게 이용될 수 있다. 기본적으로는 서버에서 실행된다.

DXML 문서는 CGI 프로그램, 서블릿, 또는 웹 서버 플러그-인을 포함하는 임의의 웹 브라우저 확장 메커니즘을 이용하여 서버에서 실행될 수 있다. 클라이언트 프로그램도 또한 DXML 함수를 직접 실행할 수 있다. DXML은 XML DOM 인터페이스를 확장하는데, 이는 XML 요소 또는 함수 반환 값을 나타내는 CDATA용 DOM을 반환하는 문서 및 요소 메쏘드를 포함함으로써 이루어진다. 그러면, 클라이언트 어플리케이션은 그들의 결과를 얻기 위하여 이들 메쏘드를 실행하면 된다. 이 메커니즘은 프로그래머로 하여금 그들의 프로그램 내에서 다양한 스크립트 언어로 작성된 DXML 문서에 수용된 업무 로직을 재사용할 수 있도록 한다.

DXML 함수가 실행될 때, 함수는 자신에게 추가의 정보를 제공하는 콘텍스트 내에서 실행된다. 함수 콘텍스트는 1) 문서 질의 파라미터, 이것의 문서 콘텍스트; 2) 실행 요소의 페이런트 요소, 이것의 동적 콘텍스트; 및 3) 요소를 포함하고, 이 함수는 자신의 사전적 콘텍스트의 멤버이다. 질의 파라미터 값은 SQL에서 호스트 변수에 의해 쓰이듯이 질의 파라미터가 콜론에 선행한다는 규약을 사용하는 DXML 함수 실행 언어에서 이용 가능하다. 함수가 스크립트로서 구현되면, 함수는 함수가 실행되는 페어런트 요소의 DOM으로 접근한다. 함수가 요소 멤버 함수이면, 함수는 요소, 요소의 속성 및 콘텐트 모델의 수용하기 위한 DOM을 포함하는 부가적 콘텍스트를 가진다. 이 경우, DXML 함수는 XML 요소를 수용하기 위한 동적 콘텐트를 계산하는 데 이용되고, 이 함수의 어떠한 호출도 XML 요소 수용 인스턴스 내에서 나타난다.

DXML를 이용하는 다른 방법은 XML/XSL 브라우저에 플러그-인을 부가하는 것인데, 이 브라우저는 클라이언트 상에서 동적 XML 문서를 실행하기 위한 "가상 머신"으로서 이용되는 것이다. 이 메커니즘은 협동 요소의 집합의 영구 상태의 스냅 샷과 같은 생생하고 원활한 문서를 지원한다. 문서에의 접근은 협동의 재개 및 문서 상태의 가능한 변경을 위하여, 수용된 DXML 함수 호출을 함으로써 문서를 "일깨우는 것"이다.

도 9는 본 발명의 양호한 실시예에 따라 도 3의 실행 함수(314)와 같은 실행 함수 내에서 이용되는 절차를 나타내는 플로우차트이다. 절차는 질의를 수신함으로써 개시된다(단계 900). DXML 문서는 소스로부터 얻어지고 질의에 기초하여 판독된다(단계 902). 소스는, 예컨대 문서를 수용하는 웹 서버 저장소일 수 있다. DXML 문서의 함수를 찾는다(단계 904). 질의로부터의 파라미터는 함수 내부로 대체된다(단계 906). 대체된 파라미터를 수용하는 함수가 실행된다(단계 908). 함수의 실행 결과는 DXML 문서내의 함수의 호출로 대체된다(단계 910). DXML 문서는 함수의 실행 후에 XML 문서로 된다. XML 문서가 반환되고 절차가 종료된다(단계 912).

도 10은 본 발명의 양호한 실시예에 따라 묘사된 다양한 콤포넌트 사이의 데이터 플로우를 도시한 도면이다. 이 실시예에서, 사용자 에이전트(1000)는 서비스 요청을 개시한다(단계 p1). 사용자 에이전트(1000)는 통상적으로 클리이언트(1002)에 위치한 프로세스이다. 사용자 에이전트에 의하여 형식화된 요구는 웹 서버에 의해 공급되는 요망 서비스를 나타낸다. 사용자 에이전트(1000)은, 예컨대 사용자 또는 타인의 어플리케이션과 같은 상이한 형태를 취할 수 있다.

이 요청은 클라이언트(1002)에 의해 처리되고, 통상적으로 통신 채널(1006)을 통해 서버(1004)로 전달된다(단계 p2). 통신 채널(1006)은 도 1의 네트워크에서 발견되는 매체와 같은 다양한 형태일 수 있다. 요청은 클라이언트(1002) 및 서버(1004) 모두가 이해할 수 있는 프로토콜을 이용하여 통신 채널(1006)을 통해 전달된다. 서버(1004)는 요청에 완전히 반응하기 위하여 백엔드 서비스(1008)를 이용할 수 있다(단계 p3). 백엔드 서비스(1008)는 서버(1004)로 결과를 반환한다(단계 p4). 본 실시예에서, 백엔드 서비스(1008)는 전래(傳來)의 프로세스 또는 시스템을 포함할 수 있다. 이 전래 시스템은 서버(1004) 또는 다른 서버에 위치할 수 있다. 요청 및 서버(1004)상의 업무 로직은 그러한 위임이 요청의 충족 또는 반응시 어떻게 일어날 것인지를 규정하는 데 이용된다.

서버(1004)는 응답을 생성하고 이것을 클라이언트(1002)로 통신 채널(1006)을 이용하여 전달할 것이다(단계 p5). 클라이언트(1002)는 그 다음 사용자 에이전트(1000)에 대한 요청을 처리할 것이다(단계 p6). 이 절차는 사용자에게 보여지는 데이터 표출을 포함한다. 이 표출은 HTML 문장을 번역하는 것처럼 쉬울 수도 있고, 계산, 연산 정렬, 또는 데이터에 대한 기타의 조작 등과 같이 복잡할 수도 있다. 대안으로서, 응답 내에서 반환되는 데이터는 클라이언트(1002)에 위치한 다른 어플리케이션에 의해 이용될 수 있다.

예를 들어, 웹 브라우저인 간단한 웹 어플리케이션으로써, 사용자 에이전트(100)는 사용자 입장에서 클라이언트(1002)를 이용하여 HTML 페이지에 대한 URL을 입력한다. 웹 라우저는 TCP/IP 상의 웹 서버인 서버(1004)로 요청을 전달하기 위해 HTTP를 이용한다. 요청은 URL을 서버상의 파일의 경로명으로 번역하는 HTTP 웹 서버에 의해 처리된다. 더 이상의 백엔드 서비스(1008)를 요하는 처리는 필요하지 않다. 웹 서버는 다시 HTTP를 이용하여 클라이언트로 응답을 전달한다. 요청한 웹 브라우저는 사용자에게 HTML 페이지를 표출한다.

약간 복잡한 실시예를 들면, 사용자는 웹 브라우저를 이용하여, SSL 접속상에서 HTML POST를 이용하는 웹 서버로 HTTP 폼을 이용하여 질의 파라미터를 제공한다. 웹 서버는 CGI 스크립트를 호출함으로써 POST 요청을 처리하는데, CGI 스크립트는 SQL 질의를 실행하기 위하여 백엔드 데이터베이스 서버의 형태인 백엔트 서비스(1008)를 연결한다. CGI 스크립트는 백엔드 데이터베이스 서버로부터 결과를 받아 그 결과를 HTML 테이블에서 형식화하며, 이 HTML 테이블은 요청한 웹 브라우저에 의해 설정된 SSL상으로 HTTP를 이용하여 클라이언트로 다시 전달된다.

더 복잡한 실시예를 들면, 사용자는 파지용 PDA상에서 실행되는 웹 브라우저를 이용하여, HTML 폼을 통하여 질의를 공급한다. 이 질의는 IIOP(Internet Inter-Object Request Broker Protocol)상에서 EJB이 가능한 웹 서버를 향해 요청으로서 전달된다. 웹 서버상의 EJB는 메시지 큐 시리즈(MQSeries)를 이용하는 다중 가상 저장(multiple virtual storage; MVS) 시스템에서 실행되는 고객 정보 제어 시스템(CICS) 어플리케이션으로 통신하는 코맨드 빈(command bean)을 호출함으로써 요청된 서비스를 공급하는 분산 업무 객체를 대표한다. MQSeries는 인터내셔널 비지네스 머신즈 코포레이션의 메세징 미들웨어인데, 이것은 프로그램이 모든 IMB 플랫폼과, 윈도우즈, VMS, 및 다양한 UNIX 플랫폼을 거쳐 서로간에 통신할 수 있도록 하는 것이다. 이 미들웨어는 프로그램이 작성되는 API와 같은 공통 인터페이스를 제공한다. 코맨드 빈은 XML 문서로서의 결과를 반환하는데, 이 문서는 클라이언트 웹 브라우저에서 실행되는 자바 애플릿으로 다시 전달된다. 클라이언트 웹 브라우저는 XSL 스타일쉬트, XML 파서, 및 XSL 프로세서를 이용하여 클라이언트 디바이스에 적합한 방식으로 문서를 번역한다.

DXML은 웹 어플리케이션의 성능을 증강하는 동시에 그들의 구현을 단순하게 하는 추가의 패턴을 도입에 대한 기회를 제공한다. 사용자 취향은 예컨대, 장치 표출 특성, 관심 영역, 사용자 인터페이스(UI), 룩-앤드-필, 세부 정도 등을 포함하는 넓은 범위에 걸쳐있다. 이러한 많은 취향은 XML 문서 접근시 상이한 XSL 스타일쉬트에 의해 간단히 취급될 수 있다. 다른 경우에, 사용자 취향에 기초한 요청되는 XML 문서의 콘텐츠를 계산한 필요가 있거나, 사용될 올바른 스타일쉬트를 계산할 필요가 있다. 서버상의 DXML 문서는 이러한 계산을 수행하는 데 사용될 수 있다. 몽상의 패턴은 사용자가 파지용 PDA상에서 실행되는 웹 브라우저를 사용하여 요청을 전송하는 것인데, 이 요청의 전송은 HTML을 이용하여 웹 서버에서 실행 가능한 DXML로 HTTP를 경유하여 요청을 전달함으로써 이루어진다. 웹 서버는 요청된 DXML 문서를 접근하고 문서를 실행하는데, 이 문서는 응답 XML 문서 콘텐트를 계산하기 위하여 사용자 취향을 접근하는 데에 사용자 에이전트에 관한 정보를 이용한다. 응답 문서는 클라이언트 웹 브라우저로 되돌려지고, 웹 브라우저는 사용자 취향에 특유한 방식으로 문서를 번역하기 위하여 XSL 스타일쉬트, XML 파서, 및 XSL 프로세서를 이용한다. DXML의 또 다른 공통적 이용은 데이터베이스로의 접근 및 질의이다. 사용자는 웹 브라우저를 이용하여 웹 서버에서 실행 가능한 DXML 문서로 HTTP를 경우하여 요청을 전달함으로써 HTML을 이용하여 요청을 제출한다. 웹 서버는 요청된 DXML 문서를 접근하여 문서를 실행하고, 요청된 데이터를 얻기 위하여 백엔드 데이터베이스 시스템을 접근하는 DXML 함수를 실행한다. 응답 문서는 클라이언트 웹 브라우저로 되돌려지고, 웹 브라우저는 문서를 번역하기 위하여 XSL 스타일쉬트, XML 파서, 및 XSL 프로세서를 이용한다. 작용이 웹상에서 더욱 보편화됨에 따라, 웹 어플리케이션은 지능형 검색 엔진에 의하여 얻어진 공개 서버스에 동적으로 결합될 수 있다. DXML은 이들 서비스를 호출하고 복합 XML 문서로 그 결과를 포획하는 데 이용될 수 있다. 검색 엔진은 광고된 서비스를 제공하는 함수를 수용하는 DXML 문서에 위치할 것이다. 함수 정의는 검색을 촉진하는 데 이용될 수 있는 키워드 같은 추가 정보를 제공하도록 확장될 것이다. 일단 클라이언트가 요망된 서비스 집합을 찾으면, DXML 문서가 서비스를 호출하고 목표 XML 또는 DXML 문서로 그 결과를 수집하는 데 이용될 수 있다. 이 문서는 XSL 스타일쉬트에 의해 처리되어 문서를 요구되는 형식으로 변환하거나, 추가의 정제를 위하여 DXML 문서로서 회귀적으로 실행될 수 있다.

6. DXML 구현의 예

도 11-14B는 본 발명의 양호한 실시예에 따라 묘사된 DMLX 구현례를 도시한 도면이다. 본 실시예에서는 도 5의 문서(500)에서 발견되는 문서 함수와 같은 문서 함수의 사용을 포함한다. 이 문서 함수는 자신의 업무 책임에 비하여 과한 급여를 받는 고용인을 검색하는 SQL 데이터베이스 질의를 위한 문서 함수 정의를 수용하는 동적 XML(DXML 또는 .dxml) 문서이다. 함수는 XML 함수 요소 내에서 정의된다. 함수 정의는 자신의 형식 파라미터, 자신들의 타입, 설명, 및 기본 값을 포함한다. 이 경우 함수 본체는 SQL로 작성된다.

도 11을 참조하면, 도 11은 본 발명의 양호한 실시예에 따라 묘사된 HTML 폼을 표출하는 데 이용되는 XML 문서의 예이다. 도 11의 XML 문서는 예시에서 함수 재사용의 편의상 도 5의 함수 정의를 포함한다. XML 문서(1100)은 도 5의 XML 문서(500)를 외부 개체(entity)로서 포함하고, 도 12A 및 12B의 XSL 스타일쉬트(1200)을 규정하여 형식 파라미터에 대한 값을 요청하기 위한 HTML 폼으로 함수를 번역한다. 이들 세 개의 문서가 함수들이 별개의 문서에 규정되어 용이하게 재사용될 수 있도록 허용하고, 함수를 분류하는 이유는 어떻게 보이느냐(viewd)에 기인한다. 이 문서는 또한 동일 문서를 상이한 XSL 스타일쉬트로 번역하는 기술에 대한 예시이기도 하다.

도 12A 및 12B는 본 발명의 양호한 실시예에 따라 묘사된, 도 11의 XML 문서의 번역에 이용되는 XSL 스타일쉬트의 예시이다. XSL 스타일쉬트(1200)는 도 3에 도시된 XSL 스타일쉬트의 더욱 자세한 예이다. XSL 스타일쉬트(1200)는 입력 파라미터 값을 얻기 위하여 XML 문서(1100)(overpaidEmployees.xml)의 번역에 이용되는 XSL 스타일쉬트이다. 본 실시예에서, XSL 스타일쉬트(1200)은 url, 구동기, 사용자, 및 패스워드 파라미터 표시하지 않는다. 다른 스타일쉬트는 어떤 다른 방식으로 함수에 대한 입력 파라미터를 표시하는 데 이용될 수 있거나, 함수는 입력을 확인하는 함수를 포함하여 기타의 함수와 함께 XML 문서에 포함될 수 있다.

도 13A 및 13B는 본 발명의 양호한 실시예에 따라 묘사된 실행 요소를 수용하는 XML 문서를 도시한 도면이다. XML 문서(1300)는 도 3의 동적 XML 문서의 예시이다. 본 실시예에서, XML 문서(1300)는 부서와 업무를 포함하는 회사 정보를 포함한다. XML 문서(1300)는 업무용 overpaidEmployees 함수를 실행하는 실행 요소를 수용한다. XML 실행 요소(DXML로 정의됨)은 섹션(1302) 내의 XML 문서에 탑재된다. 문서가 처리되면, 이 요소는 그 결과로 대체될 것이고, 그 결과는 XML로 번역된 질의의 결과를 수용하는 overpaidEmployees 요소가 될 것이다. 비록 요소들이 PCDATA(단순한 문자열)처럼 단순할 수 있지만, 모든 DXML 문서는 XML 요소를 반환한다.

도 14A 및 도 14B는 본 발명의 양호한 실시예에 따라 묘사된 도 13A 및 도 13B의 XML 문서를 표출하기 위한 XSL 스타일쉬트를 도시한 도면이다. XSL 스타일쉬트(1400)는 도 3에 이용된 XSL 스타일쉬트보다 더욱 상세한 예시이다. XSL 스타일쉬트(1400)는 XML 문서가 처리된 후 도 13A 및 13B의 XML 문서(1300)(companyStatistics.dxml)를 표출하기 위한 XSL 스타일쉬트이다. XSL 스타일쉬트(1400)는 한 업무에 대해서, 회사, 회사명 및 설명에 관한 간단한 보고서를 표시하고, 과급여 고용인에 관한 표를 제공한다. 이들 실시예에서, 서블릿은 다양한 함수를 제공하는 데 이용될 수 있다. 예를 들어, XML을 XSL 스타일쉬트를 이용하여 HTML로 번역하는 서블릿은 XML 및 XSL을 직접 다룰 수 없는 웹 브라우저를 위하여, XML을 HTML로서 번역하도록 채택된 것이다.

더욱이, DXML 서블릿은 함수 요소를 제거하고 결과되는 XML 요소 또는 PCDATA로써 실행 요소를 대체함으로써, 동적 XML 문서를 처리한다. DXML 서블릿은 많은 스크립트 언어를 지원하는 데 이용될 수 있다. 서블릿은 그 결과로써 실행 요소를 대체한 후에 원래 문서를 출력한다. DXML 서블릿에서, 실행 요소의 실제 파라미터 값은 ":variableName"을 포함하여, 그 값이 문서 질의열 중 유사하게 명명된 파라미터로부터 얻어진다는 것을 나타낸다.

발명의 효과

따라서, 본 발명은 문서에 이용되는 재사용 콤포넌트에 대한 방법, 장치, 및 컴퓨터에 구현된 명령어를 제공한다. 본 발명은 로직 또는 함수가 여러 상이한 문서에서 재사용이 가능한 방식으로 에플리케이션 또는 동적 문서를 구축하는 문서 중심적 구조를 포함한다. 실시예에서의 DXML의 사용을 통하여, 웹 어플리케이션의 복잡도가 스키마, 로직 개발, 데이터, 및 명령어 번역을 분리함으로써 감소된다. 함수는 XML이 함수 호출에 이용되는 스크립트 언어 또는 기계어(native language)를 이용하여 구현된다. 이들 함수의 호출은 함수 구현 메커니즘과는 독립적이다.

전술한 바와 같이, 본 발명의 웹 어플리케이션 개발은 마크업 언어에 작용을 부가하는 것을 허용한다. 본 발명의 메커니즘은 전술한 바와 같은 콤포넌트 모델에 기초한 활성 문서를 제공함으로써 ASP 및 JSP와 같은 템플릿 언어의 이용을 피한다. 이러한 방식으로, 문서 저작자는 함수 호출을 위해 프로그램 영역을 전환할 필요가 없다. DXML로써, 문서 저작자는 단지 XML을 이용하면 된다. 메쏘드에 대한 인터페이스 및 메쏘드의 호출은 모두 XML을 이용하여 정의된다. 함수 본체만이 그 구현에 있어서 언어 의존적이다. 더욱이, 본 발명의 DXML 함수는 문서 자체 내에 포함되기 보다는 XML 스키마 내에서 정의될 수 있다. 더욱이, IE5의 작용에서와 같은 메쏘드의 암시적 호출은 직접 함수 호출의 사용으로 회피된다. 이러한 이점은 DXML에 관하여 전술한 형식 콤포넌트 모델을 통하여 제공될 수 있는데, 형식 콤포넌트 모델은 그 내부에서 DXML이 XML을 이용하여 함수를 정의하고 실행한다. 이 메커니즘은 함수를 직접 실행하기 위하여 스크립트 언어를 채택하는 것과 대비된다. 이 방식으로, 함수 선언 및 호출이 언어 독립적이 된다.

함수는 그 범위에서 기여하는 XML 요소의 메쏘드가 될 있다. 형식 콤포넌트 모델을 이용하면 스크립트 언어의 이용을 보강하고, 문서 개발을 단순화하며, 문서 스키마 저작자, 문서 저작자, 및 문서 레이아웃 규칙 사이를 명확히 분리한다. 본 발명의 모델은 CORBA, EJB, 및 DCOM과 같은 현존하는 분산 객체 콤포넌트 모델을 통일화하고, 이들 분산 콤포넌트 모델의 서비스가 XML 문서 저작자에게 XML을 통하여 가능하도록 한다.

본 발명이 완전히 기능적인 데이터 처리 시스템의 관점에서 기술되었지만, 본 기술 분야의 통상의 지식을 가지는 자라면 본 발명의 절차는 명령어 내장 컴퓨터 판독 가능 매체 또는 그 외의 다양한 형태로 배포될 수 있고, 본 발명은 실제 배포에 이용되는 특정 타입의 신호 유지 매체에 무관하게 동일하게 적용될 수 있다는 것을 알 수 있을 것이라는 점은 꼭 주의해야 할 사항이다. 컴퓨터 판독 가능 매체의 예는 플로피 디스크, 하드 디스크 드라이브, RAM, CD-ROM 및 전송형 매체(예컨대, 디지털 및 아날로그 통신 링크, 무선 주파수 및 광파 전송과 같은 형태의 유/무선 통신 링크)를 포함한다. 컴퓨터 판독 가능 매체는 코딩된 형태를 취할 수 있으며, 이것은 특정 데이터 처리 시스템에서 실제 사용될 형태로 디코딩된다.

본 발명의 기술은 예시 및 설명의 목적이며, 본 발명을 개시된 형태로 속속히 알리거나 제한하려는 의도는 아니다. 많은 수정 및 변형이 당업자에게는 자명하다. 예를 들어, 묘사된 실시예에서 XML을 이용하는 것으로 설명되었지만, 본 발명은 기타의 마크업 언어를 이용하더라도 구현될 수 있다. 또 다른 예로서, 함수의 호출은 XML 실행 요소를 통하여 이루어질 수 있다. 또 다른 접근법은 XML 문서 언어를 SOAP처럼 이용하여 함수를 실행하는 것이다. 실시예는 본 발명의 원리, 실제 응용를 가장 잘 설명하도록 선택되고 기술된 것이고, 본 발명이 속하는 기술 분야에 통상의 지식을 가지는 자가 의도하는 특정 용도에 적합한 다양한 변형의 다양한 실시예로써 본 발명을 용이하게 이해할 수 있도록 선택되고 설명되어진 것이다. 전술한 각 실시예에서 사용된 특정 문서 언어는 DXML의 개념을 설명하기에 간단하기 때문에 선택된 것이다. 첨언하면, DXML에 채택될 수 있는 많은 다른 관련된 기술은 SOAP 및 XML 스키마와 유사하다.

(57) 청구의 범위

청구항 1.
데이터 처리 시스템에서 마크업 언어로 동적 전자 문서를 생성하는, 데이터 처리 시스템에 구현된 방법에 있어서,

마크업 언어로 된 정적 콘텐트를 동적 전자 문서에 위치시키는 단계와,

상기 마크업 언어를 이용하여 함수 호출 메커니즘을 상기 전자 문서에 위치시키는 단계를 포함하고,

상기 함수 호출 메커니즘은 함수명을 포함하고 함수 호출에 이용되는 것인 방법.

청구항 2.
제1항에 있어서,

함수명, 형식 파라미터, 및 함수 본체를 포함하는 함수 정의를 이용하여 함수를 정의하는 단계를 더 포함하는 방법.

청구항 3.
제2항에 있어서, 상기 함수는 제2 동적 전자 문서에 위치하는 것인 방법.

청구항 4.
제2항에 있어서, 상기 함수 본체는 언어를 포함하는 것인 방법.

청구항 5.
제4항에 있어서, 상기 언어는 스크립트이고, 상기 함수 본체는 스크립트 내의 명령어를 더 포함하는 것인 방법.

청구항 6.
제4항에 있어서, 상기 언어는 모 언어(native language)이고, 상기 함수 본체는 비어 있으며, 상기 함수는 동적 전자 문서 외부에서 구현되는 것인 방법.

청구항 7.

제1항에 있어서, 상기 함수 호출 메커니즘은 함수를 구현하는 객체에 대한 레퍼런스를 포함하는 것인 방법.

청구항 8.
제1항에 있어서, 상기 마크업 언어는 확장 가능 마크업 언어(XML)인 것인 방법.

청구항 9.
제1항에 있어서, 상기 함수는 스크립트로 작성된 명령어를 포함하는 것인 방법.

청구항 10.
제1항에 있어서, 상기 함수는 상기 마크업 언어에 내장된 스크립트 언어와는 다른 언어로 작성된 것인 방법.

청구항 11.
제1항에 있어서, 상기 함수는 플러그-인, 애플릿, 서블릿, 및 어플리케이션 중 하나에 구현되는 것인 방법.

청구항 12.
데이터 처리 시스템에 이용되는 전자 문서 시스템에 있어서,

함수와,

마크업 언어로 된 콘텐트와, 상기 마크업 언어로 작성된 함수 호출 메커니즘을 포함하는 전자 문서

를 포함하고,

상기 함수 호출 메커니즘의 개시에 응답하여, 상기 함수를 호출하고, 상기 함수로부터의 결과를 상기 전자 무서 내의 함수 호출 메커니즘을 대체하는 시스템.

청구항 13.
제12항에 있어서, 상기 전자 문서는 제1 데이터 처리 시스템에 위치하는 것이고, 상기 함수는 상기 제1 데이터 처리 시스템과 떨어져 있는 제2 데이터 처리 시스템에 위치하는 원격 함수인 것인 시스템.

청구항 14.
제12항에 있어서, 상기 전자 문서는 제1 전자 문서이고,

상기 함수가 위치되는, 상기 마크업 언어로 작성된 제2 전자 문서를 포함하는 것인 시스템.

청구항 15.
제12항에 있어서, 상기 전자 문서를 제시하는 데 쓰이는 스타일쉬트를 더 포함하는 시스템.

청구항 16.
분산 데이터 처리 시스템에 있어서,

네트워크와,

상기 네트워크에 연결된 클라이언트와,

상기 네트워크에 연결되고, 전자 문서를 포함하는 서버

를 포함하고,

상기 클라이언트로부터 네트워크를 경우하여 상기 서버로 요청이 전달되고, 상기 전자 문서는 상기 요청에 응답하여 실행되며, 그 결과가 상기 클라이언트로 반환되고, 그 결과가 클라이언트에 있게되는 데이터 처리 시스템.

청구항 17.
마크업 언어로 동적 전자 문서를 생성하는 데이터 처리 시스템에 있어서,

마크업 언어로 된 정적 콘텐트를 상기 전자 문서에 위치시키는 제1 배치 수단과,

상기 마크업 언어를 이용하여 함수 호출 메커니즘을 상기 전자 문서에 위치시키는 제2 배치 수단

을 포함하고,

상기 함수 호출 메커니즘은 함수명을 포함하고, 함수의 호출에 이용되는 것인 데이터 처리 시스템.

청구항 18.
마크업 언어로 전자 문서를 생성하기 위한, 컴퓨터 판독 가능 매체 내의 컴퓨터 프로그램 제품에 있어서,

마크업 언어로 된 정적 콘텐트를 상기 전자 문서에 위치시키는 제1 배치 수단과,

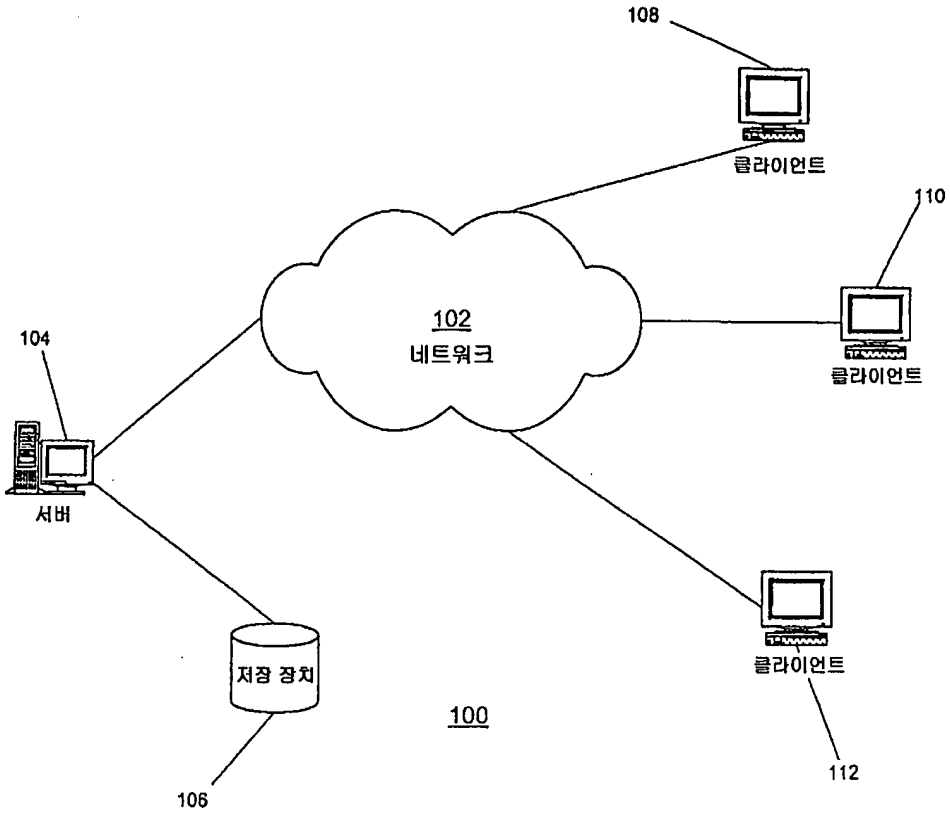
상기 마크업 언어를 이용하여 함수 호출 메커니즘을 상기 전자 문서에 위치시키는 제2 배치 수단

을 포함하고,

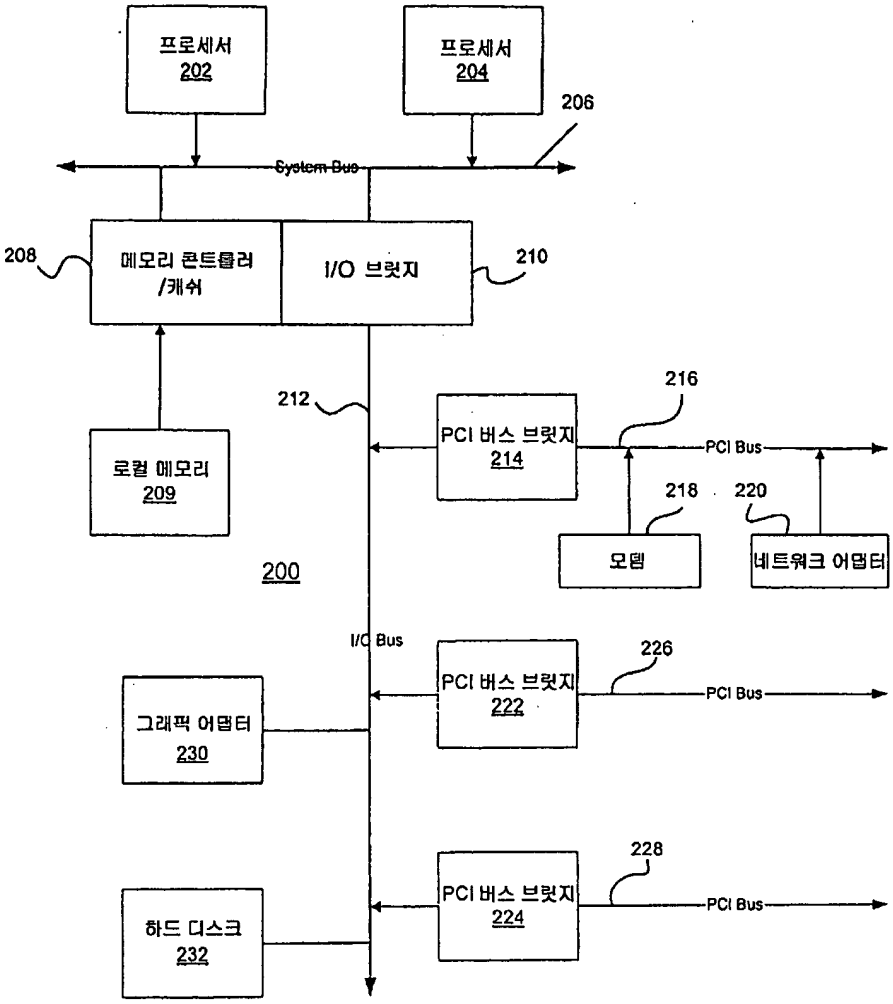
상기 함수 호출 메커니즘은 함수명을 포함하고, 함수의 호출에 이용되는 것인 프로그램 제품.

도면

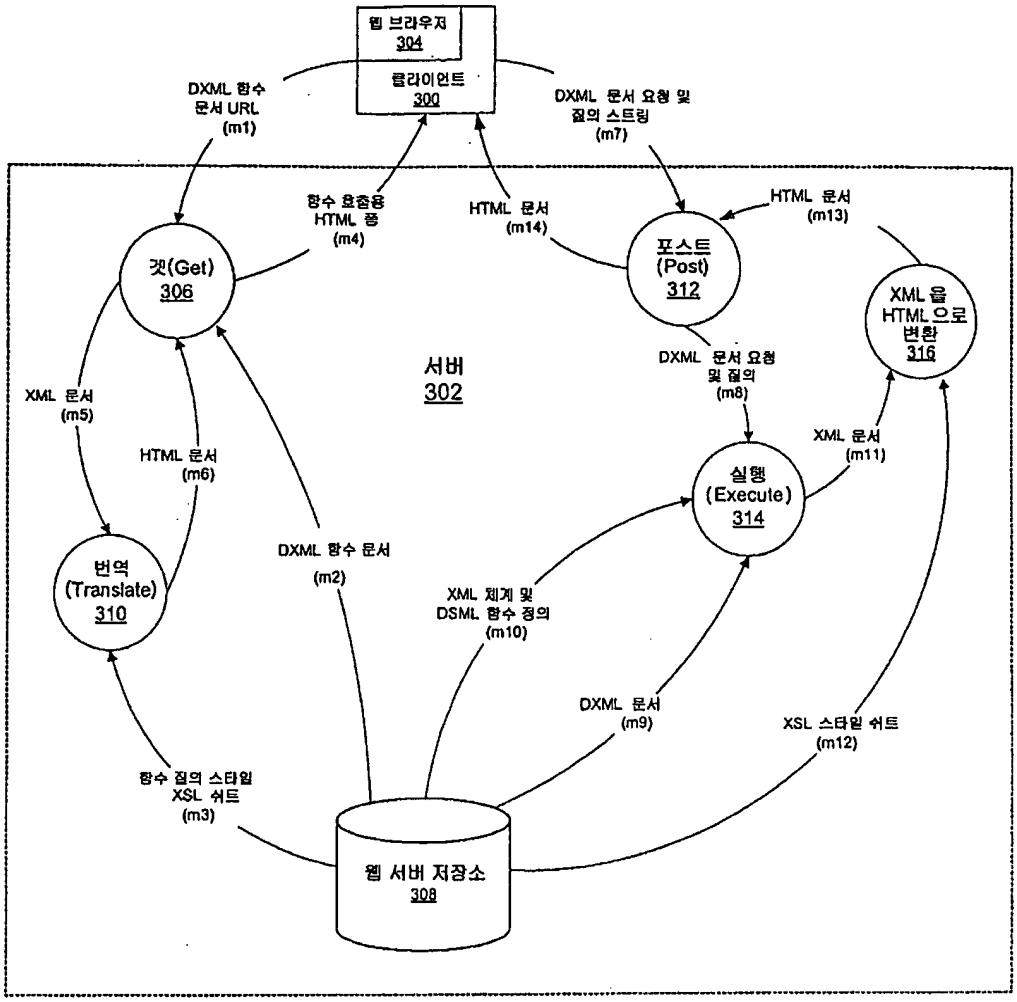
도면 1



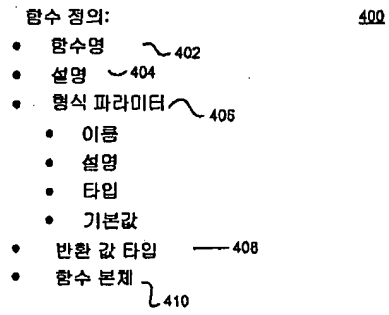
도면 2


프로세서
202
프로세서
204
206
System Bus
208
메모리 콘트롤러
/캐쉬
I/O 브릿지
210
212
216
PCI 버스 브릿지
214
PCI Bus
로컬 메모리
209
218
220
모뎀
네트워크 어댑터
200
I/O Bus
226
PCI 버스 브릿지
222
PCI Bus
그래픽 어댑터
230
228
PCI 버스 브릿지
224
PCI Bus
하드 디스크
232

도면 3


웹 브라우저
304
클라이언트
300
DXML 함수
문서 URL
(m1)
DXML 문서 요청 및
질의 스트링
(m7)
함수 호출용
HTML 폼
(m4)
HTML 문서
(m14)
HTML 문서
(m13)
겟(Get)
306
포스트
(Post)
312
XML을
HTML으로
변환
316
서버
302
DXML 문서 요청
및 질의
(m8)
XML 문서
(m5)
HTML 문서
(m6)
XML 문서
(m11)
실행
(Execute)
314
번역
(Translate)
310
DXML 함수 문서
(m2)
XML 체계 및
DSML 함수 정의
(m10)
DXML 문서
(m9)
XSL 스타일 쉬트
(m12)
함수 질의 스타일
XSL 쉬트
(m3)
웹 서버 저장소
308

도면 4


함수 정의:
400
함수명
402
설명
404
형식 파라미터
405
이름
설명
타입
기본값
반환 값 타입
408
함수 본체
410

도면 5

```
<!--Define a function to search the database for employees who
   are overpaid for their job responsibilities.

   This is an example of a DXML function that uses SQL as the
   host language. There are a number of parameters, but most
   have reasonable defaults. This is a "document" function, it does
   not belong to any particular element. Its context is defined
   by the document query parameters (the global variables), and
   the parent of the execute elements that invoke it.-->

<function
 id="overpaidEmployees"
 description="Get employees performing a job with a salaray higher than a given salary.">
 <paramdef name="url" default="jdbc:db2:sample"/>
 <paramdef name="driver" default="COM.ibm.db2.jdbc.app.DB2Driver"/>
 <paramdef name="user" default="jamsden"/>
 <paramdef name="password" default="mmjj3ch"/>
 <paramdef
  name="job"
  type="varchar"
  default="MANAGER"
  description="Job name"/>
 <paramdef
  name="salary"
  type="integer"
  default="20000"
  description="Maximum salary"/>
 <return>overpaidEmployees</return>
 <body language="SQL">
  SELECT * FROM EMPLOYEE WHERE JOB = ? AND SALARY > ?
 </body>
</function>
```

500

502

504

506

도면 6

600

```
<department name="Global Services">
  Global Services provides perform capabilities for customers.
  <practice name="Object Technology Practice">
   The Object Technology Practice provides a center of competence
   withing Global Services for developing customer solutions exploiting
   the capabilities of object-oriented technology.
   <execute id="overpaidEmployees">
    <param name="job" value=":job"/>
    <param name="salary" value="100000"/>
   </execute>
  </practice>
 </department>
```

602

도면 7

700

```
<department name="Global Services">
  Global Services provides perform capabilities for IBM customers.
  <practice name="Object Technology Practice">
   The Object Technology Practice provides a center of competence
   withing Global Services for developing customer solutions exploiting
   the capabilities of object-oriented technology.
   <execute
    href="iiop://someDomain/aCompany"
    id="overpaidEmployees">
    <param name="job" value=":job"/>
    <param name="salary" value="100000"/>
   </execute>
  </practice>
 </department>
```

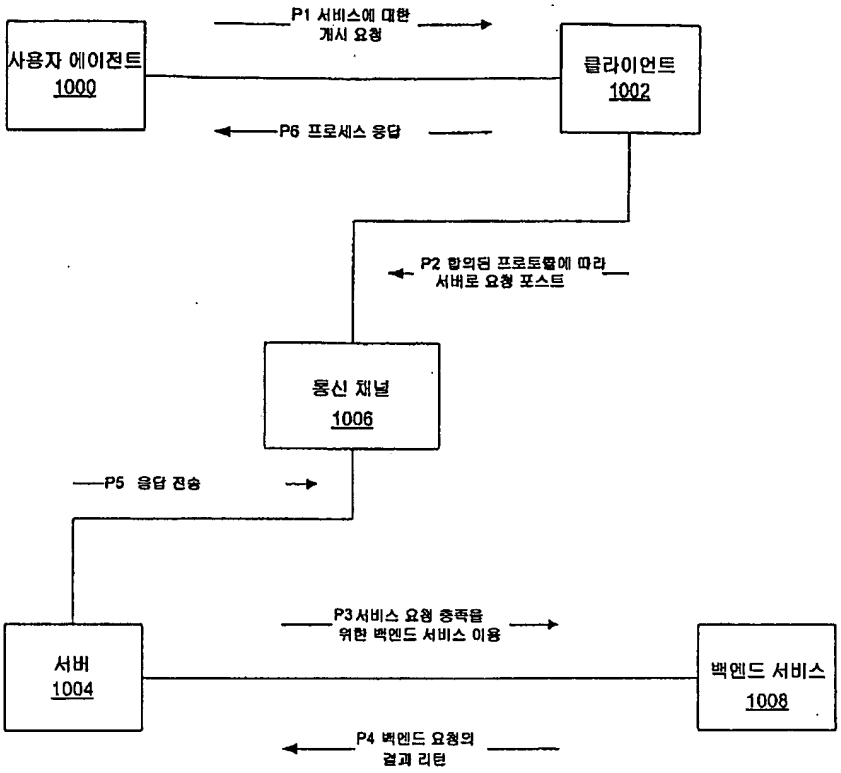
도면 8

```
<param name="salary" value="{getMaximumSalary('MANAGER')}"/>
```

도면 9


시작
DSML문서 수신 및 질의
900
DSML문서 획득
902
함수 탐색
904
질의로부터 함수로 파라미터 대체
906
함수 실행
908
DSML 문서에 결과 반영
910
XML문서로서 리턴
912
종료

도면 10



도면 11

1100

```
<?xml version="1.0" encoding="utf-8"?>
<?xml:stylesheet href="overpaidEmployees.xsl" type="text/xsl"?>
<!DOCTYPE overpaidEmployeesForm [
  <!ENTITY overpaidEmployees SYSTEM "overpaidEmployees.dxml">
]>

<!--Display a form requesting a job title and salary amount in
    order to obtain a company statistics report that includes
    information about overpaid employees.

    This document uses a stylesheet to render the included DXML
    function in order to display a form for obtaining values
    for actual parameters. The submit button on the form
    access a DXML document giving it a query string that contains
    the input parameters. This document really just provides a
    stylesheet for rendering the function.-->

<overpaidEmployeesForm>
  &overpaidEmployees;
</overpaidEmployeesForm>
```

도면 12a

1200

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xsl:stylesheet>
<xsl:stylesheet xmlns:xsl='http://www.w3.org/TR/WD-xsl'
 xmlns:html="http://www.w3.org/TR/REC-html40/"
 result-ns="html" indent-result="yes">


<!--This is a stylesheet for rendering overpaidEmployees.xml.
   It displays an HTML form requesting the input parameters:
   job and salary. The formal parameter descriptions are used
   as prompts for the fields, and the field names are the same
   as the formal parameter names. The submit button POSTs a
   request to access the companyStatistics.dxml page giving
   a query string containing the parameters.

   The url, driver, user, and password parameters (used to
   connect to the database) are not displayed in this form.-->

<xsl:template match="/">
 <HTML>
  <xsl:process select="overpaidEmployeesForm/function"/>
 </HTML>
</xsl:template>

<xsl:template match="overpaidEmployeesForm/function">
 <HEAD>
 <TITLE><xsl:value-of expr="attribute(id)"/></TITLE>
 </HEAD>
 <BODY>
   <DIV align="center" font-size="20pt" font-weight="bold">
     <xsl:value-of expr="attribute(id)"/>
   </DIV>
   <BR/>
   <xsl:value-of expr="attribute(description)"/>
   <HR/>
   <FORM method="POST" action="http://localhost:8080/companyStatistics.dxml">
     <xsl:process select="paramdef"/>
     <INPUT type="submit"/>
   </FORM>
 </BODY>
</xsl:template>
```

도면 12b

1200

```
<xsl:template match="paramdef">
  <xsl:value-of expr="attribute(description)"/><xsl:text>: </xsl:text>
  <INPUT TYPE="text" NAME="{attribute(name)}" SIZE="32" MAXLENGTH="80"/>
  <BR/>
</xsl:template>

<xsl:template match="*">
</xsl:template>

<!--Parameters we don't want to bother with-->
<xsl:template match="paramdef[attribute(name)='url']">
</xsl:template>
<xsl:template match="paramdef[attribute(name)='driver']">
</xsl:template>
<xsl:template match="paramdef[attribute(name)='user']">
</xsl:template>
<xsl:template match="paramdef[attribute(name)='password']">
</xsl:template>

</xsl:stylesheet>>
```

도면 13a

1300

```
<?xml version="1.0" encoding="utf-8"?>
<?xml:stylesheet href="companyStatistics.xsl" type="text/xsl"?>
<!DOCTYPE company [
 <!ENTITY overpaidEmployees SYSTEM "overpaidEmployees.dxml">
]>

<!--This document is an example of executing a DXML function
   in a dynamic XML document. This document can be accessed
   and executed with a simple url:

      http://hostname/companyStatistics.dxml?job=MANAGER&salary=20000

   The query parameters are substituted for the "query parameter"
   names in the execute param values. The function is executed
   and the execute element is substituted with its results creating
   a new XML document which is then rendered using a style sheet.-->
```

도면 13b

1300

```
<company name="ACME Software">
 ACME Software is a large software company specializing in exploiting
 technologies on the web. It contains a number of departments,
 each of which may have a number of practices that specialize in
 a particular area.

 &overpaidEmployees;  <!-- include the function -->

 <department name="Software Group">
  Software Group is responsible for developing software.
 </department>

 <department name="Global Services">
  Global Services provides perform capabilities for ACME customers.
  <practice name="Object Technology Practice">
   The Object Technology Practice provides a center of competence
   withing Global Services for developing customer solutions exploiting
   the capabilities of object-oriented technology.
   <execute id="overpaidEmployees">
    <param name="job" value=":job"/>
    <param name="salary" value=":salary"/>
   </execute>
  </practice>
 </department>
</company>
```

1302

도면 14a

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xsl:stylesheet>

<xsl:stylesheet xmlns:xsl='http://www.w3.org/TR/WD-xsl'
 xmlns:html="http://www.w3.org/TR/REC-html40/"
 result-ns="html" indent-result="yes">
                                                    1400
<!--Render statistics about a company including overpaid
  employees-->

<xsl:template match="/">
 <HTML>
  <HEAD>
   <TITLE>Company Statistics and Overpaid Employees</TITLE>
  </HEAD>
  <BODY>
   <xsl:process-children/>
  </BODY>
 </HTML>
</xsl:template>


<xsl:template match="company">
 <H1><xsl:value-of expr="attribute(name)"/></H1>
 <xsl:process-children/>
</xsl:template>

<xsl:template match="department">
 <H2><xsl:value-of expr="attribute(name)"/></H2>
 <xsl:process-children/>
</xsl:template>
```

도면 14b

```
                                                    1400

<xsl:template match="practice">
 <H3><xsl:value-of expr="attribute(name)"/></H3>
 <xsl:process-children/>
</xsl:template>

<xsl:template match="overpaidEmployees">
 <H4>Overpaid Employees</H4>
 <P>
  The following table lists the employees who are overpaid
  for their particular job responsibilities.
 </P>
 <TABLE align="center" border="double" cols="4">
  <CAPTION>Overpaid Employees</CAPTION>
  <xsl:for-each select="row">
   <TR>
    <xsl:for-each select="*">
     <TD><xsl:process-children/></TD>
    </xsl:for-each>
   </TR>
  </xsl:for-each>
 </TABLE>
</xsl:template>

<!--Ignore everything else-->
<xsl:template match="*" priority="-1">
</xsl:template>

</xsl:stylesheet>
```